POLITECNICO DI TORINO

Master's Degree in Biomedical Engineering

Master's Degree Thesis

"Automatic Segmentation of 3D Large Field of View OCTA Skin Volumes using Deep Learning-based Methods"

Supervisors

Prof. Kristen M. MEIBURGER

Dr. Mengyang LIU

Dr. Giulia ROTUNNO

Candidate

Lorenzo PATANÈ

Academic Year 2022/2023

Summary

The skin, as a complex organ, comprises an intricate microvasculature that is essential for its function and health. With the ability to image capillary blood flow and internal structures of the skin in a living subject, Optical Coherence Tomography-Angiography (OCTA) has the potential to offer new perspectives on the underlying causes and dynamic changes of skin conditions. This technology can be used for non-invasive monitoring of disease progression and treatment, making it a valuable tool for both diagnoses and treatment evaluation.
However, in order to make OCTA suitable for basic and clinical research in dermatology, such as the assessment of wound healing, diagnosis and treatment of Chronic Venous Insufficiency (CVI), basal cell carcinoma (BCC) and psoriasis among the others, it is necessary to reliably analyze large amounts of image data, and to compute vascular quantitative metrics which have already been used in preclinical studies and clinical applications. Therefore, accurate vessel segmentation is needed.
Manual segmentation of 3D OCTA volumes is challenging, time-consuming and prone to errors, especially for small vessels that are difficult to identify. To overcome the challenges of manual segmentation, automatic solutions have been presented, which include both traditional and deep learning methods, with a large amount of architectures proposed, mainly variants of the widely known U-Net. However, the majority of the studies on segmentation of OCTA images are related to ophthalmological applications, and only on the segmentation of the 2D maximum intensity projection of the 3D volumes.
It follows that the automatic segmentation of 3D OCTA skin volumes is an underexplored area of research.

This gap in the research and the difficulty to obtain accurate manual segmentations, highlight a need for the development of an automated segmentation approach to facilitate consistent and efficient analysis of skin microvasculature from 3D OCTA volumes.
The goal of this thesis project is to address this gap by leveraging deep learning methods to develop and evaluate an automatic segmentation algorithm for 3D OCTA volumes of skin. The dataset contained OCTA volumes acquired on healthy skin and skin affected by CVI, ensuring a high variability in the vascular structure of the data. The high-resolution acquisition system allowed to obtain 512x512x96 pixels large volumes, comprising a field of view (FOV) of 10x10 mm in the lateral and 1 mm in the axial direction. The total 214 sample volumes were split into training set, validation set and test set with a 50:25:25 ratio.

A deep learning-based automatic segmentation algorithm was then proposed, which uses three different model architectures, namely the 3D U-Net, 3D Res-Unet and 3D Dense-Unet, trained on OCTA volumes, and combine them with different Ensemble Learning methods with the goal to obtain the most possible connected vascular networks in the segmentations. Different loss functions (Dice, Jaccard and a recently proposed connectivity loss) were also evaluated, with data augmentation and a vesselness filter on the training set, to obtain the best model for the final testing.
Finally, two post-processing techniques were applied to the predicted segmentations to further improve the connectivity of the segmented vascular networks, through the removal of small, isolated objects and a closing operation, widely used in image processing to fill gaps and holes.
The algorithm's performances were then evaluated against the segmentations obtained using the Amira semi-automatic software.

The metrics obtained reach Dice Scores higher than 80% for 60% of the samples, with the presence of few outliers with low metric values, more often due to an inaccurate reference segmentation, rather than a low-quality automatic segmentation.
Moreover, the two post-processing techniques applied have shown to be effective in providing more connected vessels, in some cases even better than the reference segmentations.
The results are promising considering the FOV of the volumes used, and the variability in the data.
Future works might include the automatic classification of each volume based on the computation of vascular metrics, to further validate the proposed automatic segmentation methods.

Index

1. Introduction to OCTA and its Applications

Optical coherence tomography angiography (OCTA) is a non-invasive imaging technique that utilizes the principles of optical coherence tomography (OCT) to visualize the microvasculature in various tissues. OCTA uses a low-coherence interferometry to obtain high-resolution, 3D images of blood vessels without the need for injection of dyes. The operating principle exploits the fact that moving red blood cells within vessels will cause a decorrelation of the backscattered light from the tissue between different scans repeatedly taken at the same position, opposed to the static tissue around that will not give any decorrelation values. By analyzing the decorrelation, it is possible to create angiograms of the microvasculature, which provides information about the blood flow, vessel density, and perfusion. With its high resolution and non-invasive nature, OCTA is becoming a valuable tool in the diagnosis and management of various ocular and skin diseases.

In the past years, a wide variety of clinical applications using OCTA images has been reviewed in literature, particularly in ophthalmology for the diagnosis of retinal diseases, such as age-related macular degeneration (AMD) [1], diabetic retinopathy (DR) [2], retinal vein occlusion (RVO) [3] and others. With the increasing availability of OCTA systems, more and more studies have been conducted to investigate its diagnostic accuracy compared to traditional imaging modalities. For example, in [1] OCTA has been compared to Dye Angiography in the detection of AMD and the authors showed how its performances were aligned with the gold-standard method, providing therefore an alternative non-invasive method for diagnosis of AMD.

To better use the information offered by OCTA, the blood vessels in the OCTA volumes must be accurately segmented, so that different parameters useful to clinicians for disease detection can be computed, and a large amount of images can be reliably analyzed.
Manual segmentation is a time-consuming process, especially in tasks that involve the segmentation of very small vessels in complex vascular network, requires a high level of expertise, and is also prone to errors. This process becomes unfeasible for large datasets. Furthermore, manual segmentation can be affected by inter-and intra-observer variability, which can lead to inconsistent results. These reasons explain the need for an automatic segmentation of OCTA volumes.

However, due to the low contrast and high noise level of OCTA volumes, as well as the complex and varied vascular structure both in the retina and the skin, this is a difficult process. Many automatic solutions have been presented in the recent years for segmenting OCTA volumes and overcome these challenges.

In [4], the authors analyzed and compared the most commonly used methods for segmentation of OCTA images that have been proposed in literature and provided a useful and comprehensive review of the different approaches.

In their article it can be seen that the most common segmentation technique in OCTA images is thresholding, either global or local, which differ on the method used to choose the threshold that will determine which pixels belong to the object and which do not. In global thresholding a single threshold is selected for all the pixels of the image, while in local thresholding the threshold is set individually for each pixel depending on some characteristics of its neighborhood.

Following close the thresholding techniques, deep learning methods are also among the most used techniques to segment OCTA images, and the most promising, as the results in eye vasculature segmentation in [5, 6] show. Specifically, convolutional neural networks (CNNs) have been widely used due to their ability to learn feature representations from raw data and handle noise and artifacts in OCTA images, leading to more reliable and robust segmentation results. Despite these strengths, the main limitation of deep learning-based segmentation is the requirement for a large amount of labeled data, which, as already discussed, is difficult to obtain given the challenges of manual segmentation of 3D OCTA volumes.

It follows that, despite the great amount of studies on OCTA images and their segmentation methods, there is still a limited amount of research that directly segments 3D volumes, which is important for a more comprehensive understanding of the vasculature, both in the eye and the skin. From the 56 articles analyzed in [4] only 8 studies used 3D data, although 2 of these provide a 2D output segmentation.

Additionally, there are very limited studies (only 2 in the mentioned article) focusing on segmentation of OCTA volumes for dermatology which is a promising field for OCTA. This is also due to the lack of available databases for dermatological applications, given that OCTA in dermatology is still a research field. This indicates that there is still much room for growth and development in the field of OCTA volumes segmentation, particularly in the areas of 3D segmentation and dermatology applications.

Developing an automatic segmentation algorithm for 3D OCTA skin volumes might be very useful, given the increasing number of skin diseases that have been showed to be possible to image with OCTA, as described in [7], where the authors reported the many pathological skin conditions imaged with OCTA in the recent years, which include among the others psoriasis, eczema, basal cell carcinoma (BCC), scars, acne (Table 1.1).

Year	**Last Author**	**Location**	**Feature**	**Journal**
2010	R. Wang	Palm	First demonstration	OE
2011	M. Leahy	Forearm	---	BOE
2011	R. Wang	Antecubital area	Psoriasis	LSM
2012	R. Leitgeb	Forehead, forearm, cheek	Eczema, BCC	BOE
2013	D. Sampson	Forearm, thigh	Scars	JBO
2014	R. Wang	Hand	Laser burn	JBO
2015	R. Wang	Face	Acne	LSM, AO
2015	R. Wang	Finger	Inflammation	JBP
2016	A. Tosti	Nail	Psoriasis, 16 patients	SAD
2016	R. Wang	Forehead, arm, chest, areola	Nevi, acne, 7 day	SRT, SR
2016	R. McLaughlin	Forearm	Heat stress, 8 subjects	JAP
2016	R. Leitgeb	Palm, finger	---	BOE
2016	R. McLaughlin	Abdomen, arm, thigh	Burn scar, 7 patients for 7 days	BOE, JBP
2016	G. Jemec	Various	35 subjects	MR
2016	J. Welzel	Various	Atlas of diseases	D
2017	R. McLaughlin	Various	Scar under treatment, 7 patients 20 weeks	JBP
2017	G. Pellacani	Face	Acne, 31 patients, 15 for 60 days	JEADV
2017	R. Wang	Various	15 subjects	SRT
2017	J. Welzel	Various	Atlas of diseases	DT
2017	M. Ulrich	Various	18 AK, 12 BD, 24 SCC patients	JEADV
2018	R. Wang	Residual limb	Reactive hyperaemia	SRT
2018	S. Matcher	Various	Eczema 32 patients	BOE
2018	R. Wang	Various	Atlas of diseases	LSM
2018	R. Wang	Hand	Wound healing for 44 days	QIMS
2018	G. Jemec	Various	280 subjects	ED
2018	R. Wang	Upper arm	Blood collection, 19/21 subjects	LSM
2018	G. Jemec	Forearm	Histamine induced wheals, 10 subjects	SRT
2018	G. Jemec	Various	13 AK, 22BCC, 17 SCC and 81 patients for subtypes of BCC	SRT, ED

Table 1.1: *Dermatological OCTA studies in recent years, table adapted from [7].*

It has been shown also in [8] that OCTA with its ability to image capillary blood flow and structural features in living skin has the potential to provide new perspectives into pathophysiology and dynamic changes of skin diseases, improving diagnoses and non-invasive monitoring of disease progression and treatment efficacy.

Therefore, given the difficulties of manual segmentation and the limited amount of research in the field of 3D OCTA volume segmentation in skin, this thesis study aims to develop a new automatic algorithm to improve the accuracy and efficiency of the segmentation process. In particular, in this study deep learning techniques have been used to segment 3D OCTA volumes of the skin, with the goal of providing clinicians accurately segmented volumes that can be useful for monitoring of disease progression and treatment, and to start filling the gap in the field of automatic segmentation of 3D OCTA volumes in dermatology.

1.1 Optical Coherence Tomography (OCT)

Optical coherence tomography (OCT) is a powerful imaging technique that rapidly emerged in the last decades thanks to its characteristics that allow to obtain non-invasive, in vivo, high-resolution, cross-sectional images of biological tissues. OCT systems can achieve axial and lateral resolutions of a few micrometers [9,10,11].

It is particularly suited to ophthalmic applications and other tissue imaging requiring micrometer resolution and millimeter penetration depth. Mainly used in ophthalmology to obtain detailed images of the retina, recently, it has also been used in interventional cardiology to help diagnose coronary artery disease [12], and in dermatology to improve diagnosis [13].

Optical coherence tomography is based on low-coherence interferometry, that measures the echo time delay and intensity of backscattered light by comparing it to the incident light [14], and typically employs near-infrared light. The use of relatively long wavelength light allows it to penetrate deeper into the scattering medium.

OCT measurements were originally performed using a Michelson type interferometer, as illustrated in Figure 1.1, in which light is split into two arms – a sample arm (which leads to the blood vessels) and a reference arm (which usually leads to a mirror). Only when light from both arms has traveled the "same" optical distance, meaning that the difference is less than a coherence length, the combination of reference light from the reference arm and reflected light from the sample arm results in an interference pattern. By scanning the mirror in the reference arm, a reflectivity profile of the sample can be obtained (this is *Time Domain OCT*), called an A-scan, and it contains information about the spatial dimensions and location of structures within the analyzed sample. A sequence of these axial depth scans can be laterally combined to create a cross-sectional image (B-scan) [14]. Figure 1.2 shows an example of B-scan of the skin of a human leg obtained with a lab-built OCT system.

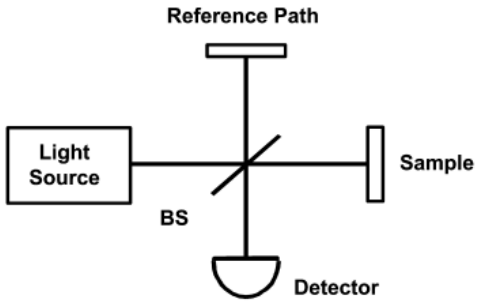


Figure 1.1: *Simple scheme of an OCT system based on a Michelson interferometer [14].*

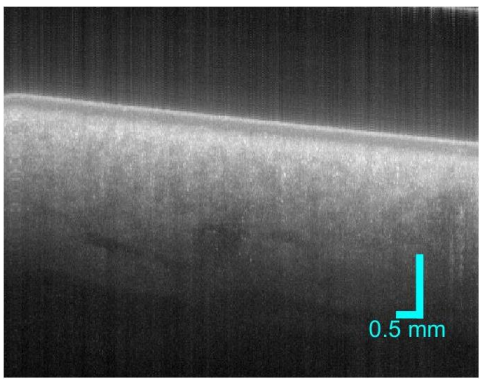


Figure 1.2*: OCT B-scan obtained with a lab-built OCT system on a healthy leg.*

After Time Domain OCT (TD-OCT), two more techniques have been developed in the recent years, improving TD-OCT characteristics:

- *Spectral Domain OCT* (SD-OCT): this technique uses a broadband light source. Unlike TD-OCT, which measures the time delay between the two beams, SD-OCT uses a spectrometer to analyze the light reflected. It separates the light into the different wavelengths and a detector measures the intensity of the light at each wavelength, and then through a Fourier transform a cross-sectional image can be reconstructed. SD-OCT has higher resolution and acquisition speed than TD-OCT (up to ~ 5 μm and 50k scans per second, against the ~10 μm and 400 scans per second of TD), which allows the implementation of OCTA.
- *Swept Source OCT* (SS-OCT): it still uses a broadband light source, but differently from SD-OCT, which uses a fixed wavelength light source and then is split at the spectrometer level. In SS-OCT the light source rapidly sweeps through a range of wavelengths, allowing for even higher imaging speed (over 100k scan per second) and resolution.

In dermatology, OCT has two main peculiarities compared to other standard techniques commonly used: compared to dermatoscopy, which provides a high-resolution en face view of the skin surface, OCT can provide cross-sectional imaging revealing tissue morphology until 1 to 2 millimeters in depth. Additionally, in comparison to the gold standard histology, OCT's non-invasiveness enables periodic imaging sessions to track the progression of pathogenesis and treatment without the need to collect tissue samples, avoiding the risks of adverse effects and scarring [9].

However, despite being established as a method to produce detailed images of the retina morphology, the OCT alone is not the best option for blood vessels imaging. That is where a functional extension of OCT, the Optical Coherence Tomography Angiography (OCTA) comes in hand.

1.2 Optical Coherence Tomography Angiography (OCTA)

Optical Coherence Tomography Angiography is a new non-invasive functional extension of OCT, which allows to obtain high-resolution volumetric angiograms providing vasculature information by using motion contrast imaging. Combining its information with the ones obtained from OCT is possible to obtain both anatomical structure and vascular structure [15,16]. Compared to other techniques such as Fluorescein Angiography, OCTA has the advantage of not requiring any dye injection, and is still able at the same time to obtain high-resolution images.

The working principle of OCTA is to detect blood flow-induced changes between successive B-scans acquired at the same sample location. Flowing red blood cells causes more variation in the OCT signal between repeated scans than static tissue. Multiple approaches have been described for quantifying this change through assessment of the intensity, phase, or intensity and phase of the OCT signal.

In OCTA, an A-scan is a unidimensional axis scan that scans the tissue in depth, along the axial direction z. A B-scan is then obtained by acquiring many parallel A-scans, while the light beam is scanned along the fast scanning direction x. In the end, the volumetric angiogram is obtained by sequentially acquiring many B-scans, along the slow scanning direction y (Figure 1.3).

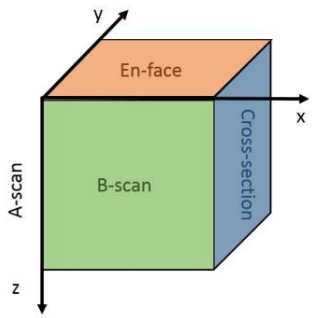


***Figure 1.3**: Schematic representation of data in OCTA.*

Figure 1.4 shows a simple scheme of how an OCTA volume is created.

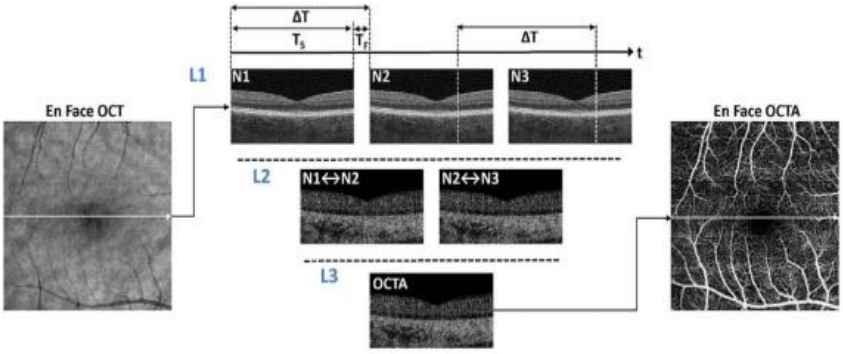


***Figure 1.4**: Simple scheme of how OCTA works. Repeated B-scans (N1, N2, N3) are acquired at the same position (L1), decorrelations between scans are calculated between successive B-scans (L2) and combined into an OCTA cross-sectional image. This procedure is repeated in the direction perpendicular to the B-scan to obtain the volumetric data [17].*

The dissimilarity between consequent B-scans in this work was obtained through the intensity-based OCTA algorithm, in which the motion contrast volume is obtained by averaging over pairwise differences of subsequent logarithmically scaled intensity tomograms taken at the same position. Compared to the phase based, and complex based methods, the intensity-based method is simpler and it has low sensitivity to phase noise, which is present if the light source is not stable.

2. Segmentation Techniques in OCTA

Accurately segmenting vessels in OCTA images is fundamental, but manual segmentation can be challenging due to the complexity of vascular networks, leading to errors even from experts and inter-intra variability errors. Moreover, OCTA volumes can be affected by different artifacts, with the most common ones being projection artifact and movement artifact. The first affects the visualization of deeper vascular networks and is caused by fluctuating shadows cast by flowing blood cells in the more superficial vessels. The shadowgraphic projection results in variation of both amplitude and phase and can be picked up by most OCT angiography algorithms as false flow. The second occurs when there is patient movement during the acquisition and cause the image to be blurred and distorted. Factors that must be considered for the movement artifact are the acquisition time, which if longer increases the difficulty to stay completely still for the patient, and age or medical conditions of the patient, which can increase the likelihood of motion artifact in presence for example of tremors or others movement disorders. While the motion artifact can be overcome with relatively simple motion correction algorithms, the projection artifacts require more complex processing algorithms to obtain a cleaner image.

To address these issues, various segmentation techniques have been developed, with traditional thresholding-based techniques and deep learning-based approaches being among the most popular.

2.1 Traditional Methods

Thresholding methods have been widely used in the field of image processing as a simple and effective way to segment objects of interest in an image. These techniques are based on setting a certain intensity threshold that separates the objects of interest from the background. In the context of OCTA, thresholding methods can be used to segment blood vessels in the volume by setting a threshold on the intensity of the pixels representing the vessels [9,18].

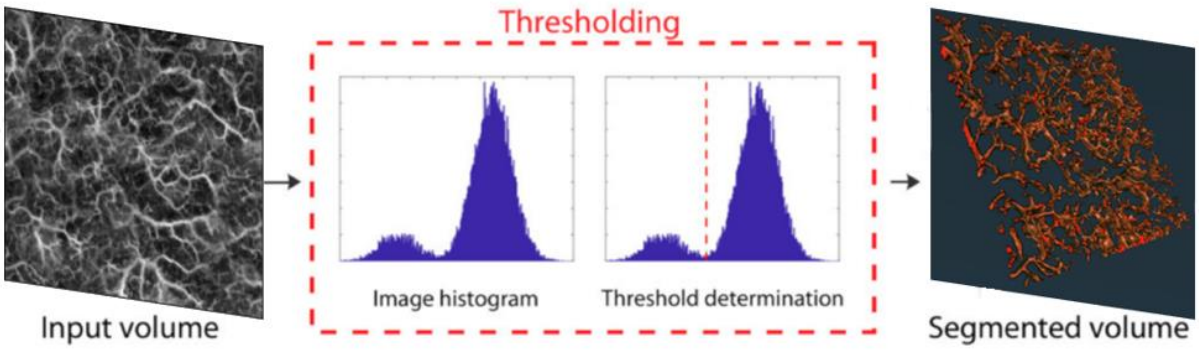


Figure 2.1: *example of thresholding.*

There are different techniques that can be used to set the threshold, which can be divided in two main categories:

- *Global thresholding*: a single threshold value is set for all the pixels of the image to split the vessel pixels from the background. The threshold is determined by selecting a pixel intensity value that maximize the separation between the two classes. This value can be determined manually, by trying different values, or automatically, using statistical methods such as the Otsu's method, one of the most used, which sets a threshold that maximizes the variance between the two classes.

In practice, global thresholding can provide a simple and efficient segmentation method for blood vessels in OCTA images. However, it may not be suitable for images with varying contrast and brightness since the threshold value may need to be adjusted for each image.

- *Adaptive thresholding*: is more suited for images with non-uniform illumination or varying local contrast. It applies a different threshold value for each pixel or local region of an image based on some local characteristics, such as the mean or median values of the local vessel intensities.
  Beside these two simpler ones, there are many other advanced methods for adaptive thresholding, which differ in the local characteristics considered.

These methods are typically fast and easy to implement but can also suffer from limitations such as sensitivity to noise, artifacts and contrast variations, which are very common in OCTA images.

2.2 Deep Learning Methods

Deep learning is a subset of machine learning that uses artificial neural networks to learn representations of data through multiple layers of non-linear transformations. It is characterized by its ability to automatically learn and extract meaningful features from raw input data, without the need for handcrafted feature engineering. Deep learning has revolutionized many industries, including healthcare, where it has been used for a wide range of applications, with the most promising ones being medical image analysis, which includes tasks such as image segmentation and classification.

Image segmentation is particularly important for many clinical applications, as it can be used to identify and delineate structures of interest within an image, such as tumors, blood vessels, or organs. Traditional methods for image segmentation often require handcrafted features, which can be time-consuming and prone to errors. In contrast, deep learning methods can learn to segment images automatically, directly from raw input data. This is accomplished by training a deep neural network on a large dataset of labeled images, where the network learns to identify and extract the relevant features from the images. Once the network is trained, it can be used to segment new, unseen images with high accuracy and efficiency. Deep learning-based image segmentation has been applied to a wide range of medical imaging modalities, including magnetic resonance imaging (MRI) [19], computed tomography (CT) [20], and OCTA, with promising results.

These methods are based on artificial neural networks, which are designed to automatically learn and extract features from the input data, enabling in this case the segmentation of complex and dynamic vascular structures. Figure 2.2 shows an example of retinal OCTA image segmentation with deep learning.

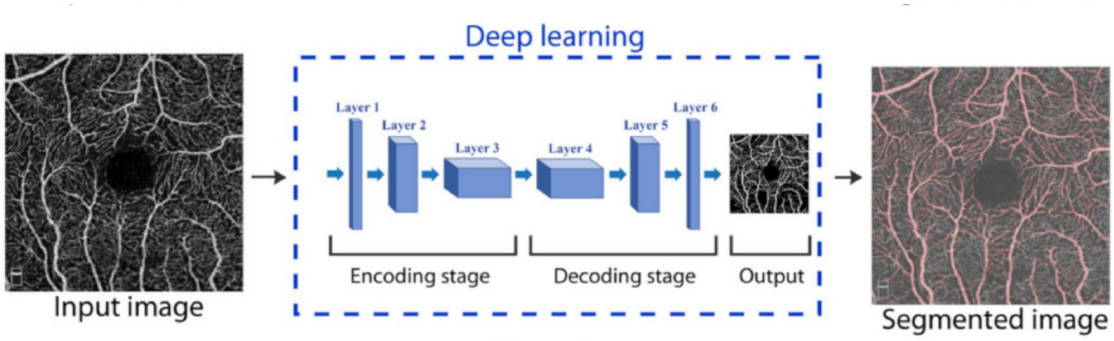


***Figure 2.2**: Example of retinal OCTA image segmentation using deep learning [4].*

Deep learning approaches have demonstrated high accuracy and robustness in segmenting vessels in 2D retinal OCTA images, surpassing the performance of traditional thresholding methods.

Some popular deep learning techniques for OCTA image segmentation include Convolutional Neural Networks (CNNs), U-Net and its variants, and many custom architectures. These models have been widely evaluated and demonstrated state-of-the-art performance on various 2D datasets, outperforming traditional thresholding methods in terms of accuracy, robustness, and speed. Despite their remarkable performance, deep learning methods for OCTA image segmentation still have some limitations, such as the need for large, annotated datasets and the high computational cost of training and inference. These limitations are even more prominent in the case of 3D OCTA datasets. Therefore, the development of new deep learning models and techniques that can address these challenges is an active area of research in the field.

In this work different architectures have been evaluated on a 3D OCTA dataset, particularly a classic 3D U-Net, and some variants of 3D U-net using different widely used CNNs as backbones, namely the ResNet-34 and the DenseNet-169. The architectures will be discussed in detail in Section 4.

3. Materials

The dataset used in this work is comprised of 214 OCTA volumes of size 512x512x96 pixel, each with the corresponding binary ground truth, obtained by segmenting the OCTA volumes on Amira [21], a software that allows to semi-automatically segment 2D/3D images by providing many tools such as thresholding, removal of "islands" (certain quantity of pixels close to each other, configurable by setting the size of the connected region), smoothing of the contours, and also the option to manually delete some pixels by passing over them with a brush.

The volumes comprise a Field of View (FOV) of 10x10 millimeters in the x-y dimensions, in the en-face view, and around 1.0 millimeters along the z axis, the axis of depth in the tissue. The use of a FOV of 10x10 mm is advantageous for OCTA skin volumes due to its ability to capture a larger surface area of the skin with high resolution, providing a more comprehensive and detailed analysis of the microvasculature compared to the traditionally used smaller FOV in literature. This larger FOV enables a more complete understanding of the vascular network and its relationship with skin pathologies, facilitating more accurate diagnosis and monitoring of disease progression. On the other hand, the larger FOV leads also to a more challenging task, compared to the segmentation of smaller FOV images in ophthalmology, where usually a 3x3 or 6x6 FOV is enough.

The volumes have been acquired from 58 patients, 29 men and 29 women ranging from 22 to 90 years old. The dataset comprises both healthy samples and samples with different stages of Chronic Venous Insufficiency (CVI), from the first to the sixth and last stage.

CVI is a medical condition that affects the veins in the legs, leading to poor blood flow and the accumulation of blood and fluid in the legs. The different stages in which CVI can be classified have different clinical features and lead to modifications in the vascular structure and morphology, which include telangiectasias (spider veins, small, dilated veins near the skin surface), reticular veins slightly larger than spider veins, varicose veins (enlarged and twisted veins protruding from the skin surface), edema and finally ulcers.

This way the dataset used presents a high variability in the vascular structure, with vessels diameters ranging from tens of μm to few mm in the bigger varicose veins, which leads to an even more challenging task.

3.1 OCT System Setup

The OCT system used to acquire the OCTA volumes is the one shown in Figure 3.1 and described in detail in [9][15], it employs a swept source with a central wavelength of 1310 nm and bandwidth of 29 nm, running at a sweep repetition rate of 200 kHz. The power incident upon the sample is 5.8 mW. The data are acquired by a digitizer at 400 MS/s. The lateral resolution is 54.64 um, while the axial resolution is 26.86 um in air and 18.52 in tissue. The field of view is 10x10 mm, and the penetration depth is almost 2 mm.

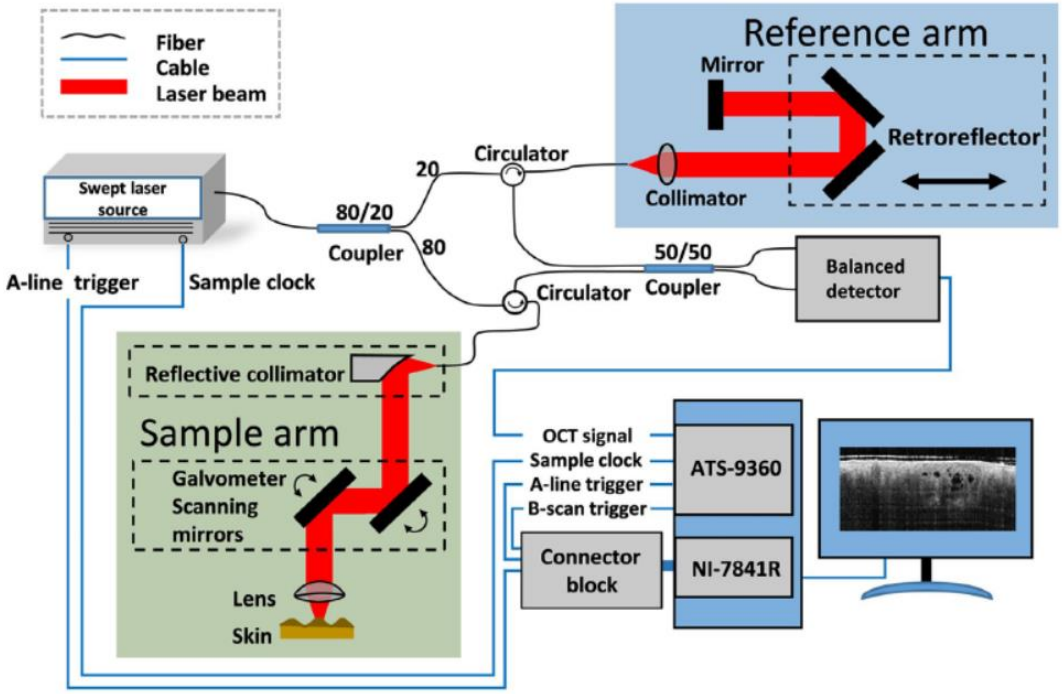


***Figure 3.1**: Scheme of the OCT system used to acquire the OCTA volumes [9].*

3.2 Data Acquisition

The data acquisition has been done with the previously described system located at the Center for Medical Physics and Biomedical Engineering at the Medical University of Vienna. The procedure followed to acquire the OCTA volumes from each patient skin is described below.

After the optic laser has been turned on, together with the galvanometers, the laser is calibrated using a custom software made available by Insight Photonic Solutions, Inc, U.S.

When the laser is ready to be used, the patient is asked to lay on the bed, to ensure he is as still and comfortable as possible. Thanks to the customized OCT system and the possible use of a vacuum cushion, more positions of the skin, usually on the legs, can be imaged by simply moving the cart on which the probe is placed.

Then a 1mm thick glass is placed between the skin and the probe to reduce the hyper-reflection from the skin surface, and some distilled water can also be used for a better refractive index matching.

The data acquisition was performed using LabVIEW, a graphical programming environment developed by National Instruments Corporation. The acquisition of one OCTA volume takes around ten seconds usually, and the quality of the acquisition can be visually checked before acquiring by watching the OCT B-scan shown in the LabVIEW interface, software used to acquire the data, and eventually different adjustments can be done before acquiring, such as the beam tilt to reduce excessive intensity reflected, the change of the focus, slight power adjustments.

3.3 Data preprocessing

After the OCT B-scans are acquired, they are processed in MATLAB, in order to create the OCTA volumes. To obtain the OCTA volumes, an intensity-based OCTA algorithm is applied, in which four B-scans successively taken at the same position are averaged, so that it is possible to visualize only what is not static in the tissue, that is blood flowing in the blood vessels. In Figure 3.2 are shown the en-face views of the Maximum Intensity Projection at two different depths interval and a B-scan of an OCTA volume,.

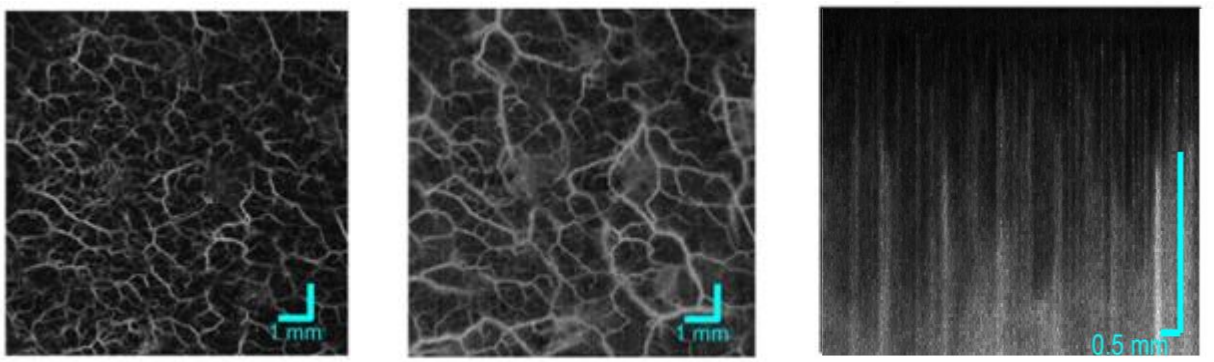


***Figure 3.2**: From left to right: en-face MIP of OCTA image in the depth range 0.192-0.575 mm, en-face MIP of OCTA image in the depth range 0.575-0.958 mm, example of OCTA B-scan.*

Now that the OCTA volumes are created, they are further processed, by firstly removing if presents motion artifacts (Figure 3.3), bright lines caused by patient movement during acquisition or by power fluctuation of the laser source. This is achieved by dividing each B-scan at different positions of the volumetric angiogram over its mean value.

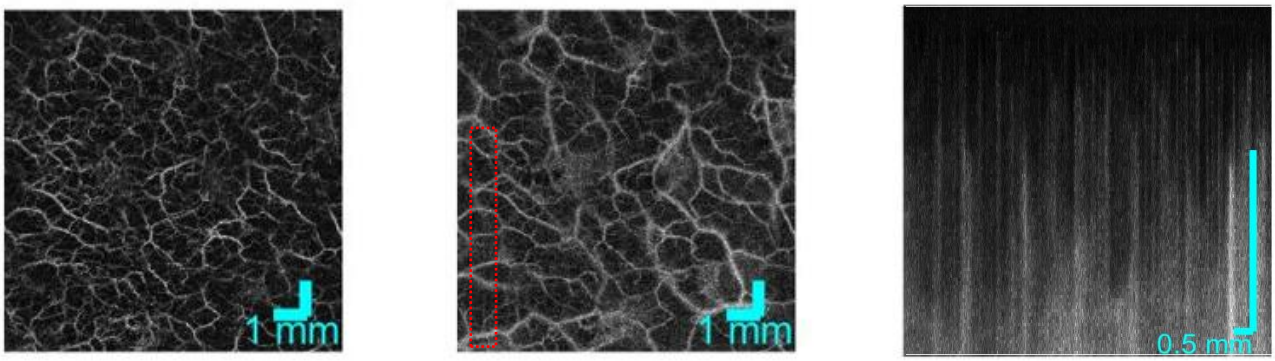


***Figure 3.3:** From left to right: en-face MIP of OCTA image in the depth range 0.192-0.575 mm, en-face MIP of OCTA image in the depth range 0.575-0.958 mm, example of OCTA B-scan. These are the images after motion correction. Although this volume was not affected by motion artifact, in the rectangle area the slight effect of the correction can be seen compared to the same area in Figure 3.2.*

Then, an algorithm for removal of projection artifacts is applied (Figure 3.4), using the step-down exponential filtering method [22] where the value of gamma determines how much information will be removed, hence it is important to not use big values, otherwise also real vessels will be removed.

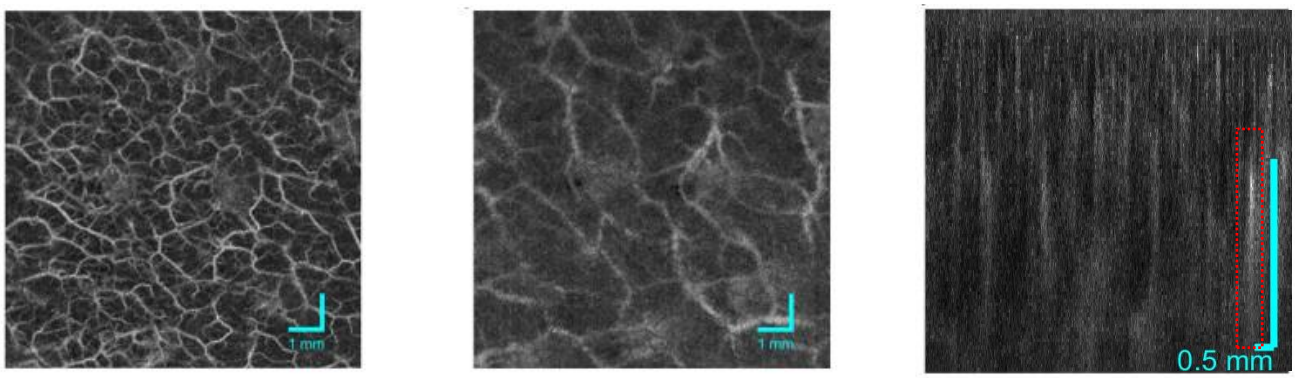


Figure 3.4: *From left to right: en-face MIP of OCTA image in the depth range 0.192-0.575 mm, en-face MIP of OCTA image in the depth range 0.575-0.958 mm, example of OCTA B-scan. These are the images after projection artifact attenuation.*

It can be observed from Figure 3.4 that in the central image, depicting the deeper interval, many vessels belonging to shallower layers have been removed, and it is even more noticeable by looking at the elongated line highlighted in red, which is less evident than in the previous image in Figure 3.3.

Finally, a median filter with a kernel size of 3x3x3 is applied to denoise the volumes, followed by a min-max scaling normalization to have only values in the range [0.0, 1.0] and a contrast enhancement, to better visualize vessels (Figure 3.5).

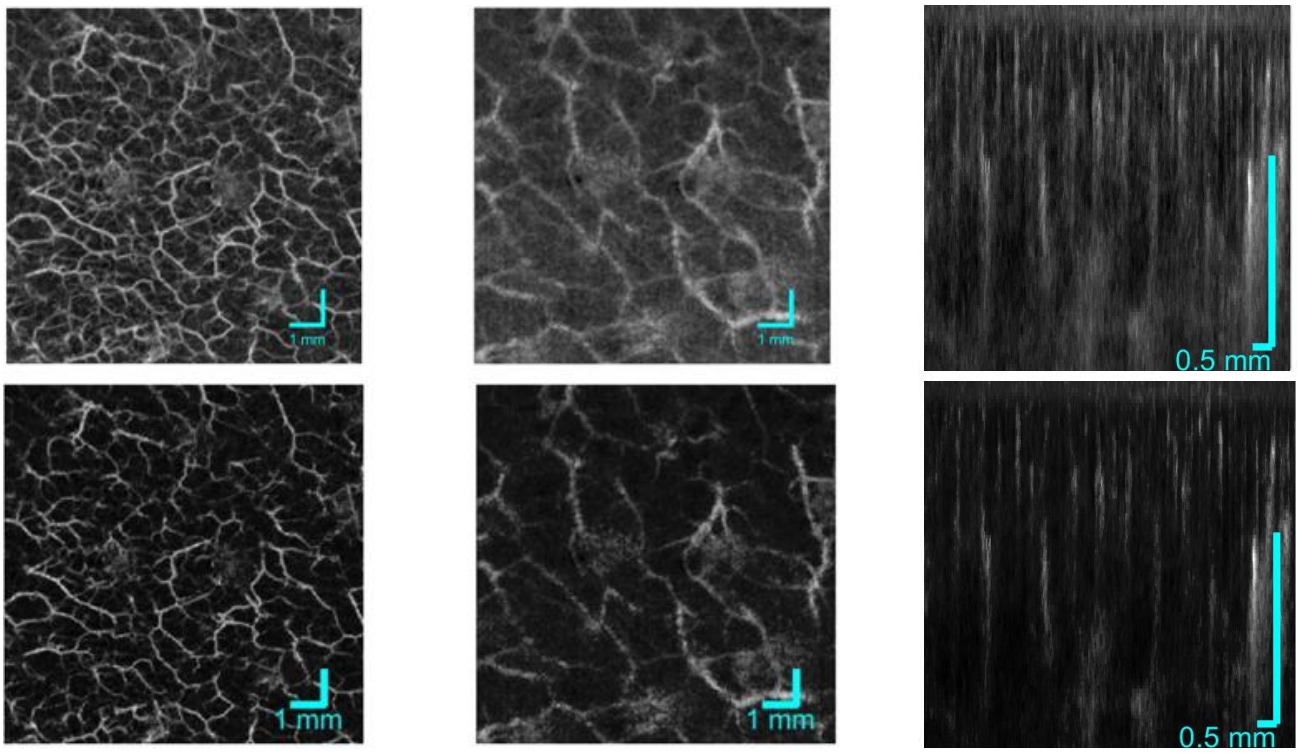


Figure 3.5*: Top row – images after median filtering. Bottom row – images after contrast enhancement. From left to right: en-face MIP of OCTA image in the depth range 0.192-0.575 mm, en-face MIP of OCTA image in the depth range 0.575-0.958 mm, example of OCTA B-scan.*

The processed OCTA volumes are then segmented on Amira and used as inputs to the different network architectures. Figure 3.6 shows an example of OCTA volume segmented in Amira, and the en-face view of its

MIP. The operations done in Amira usually are thresholding, smoothing of the edges, removal of small isolated objects smaller than a certain size and manual removal of artifacts with a brush tool.

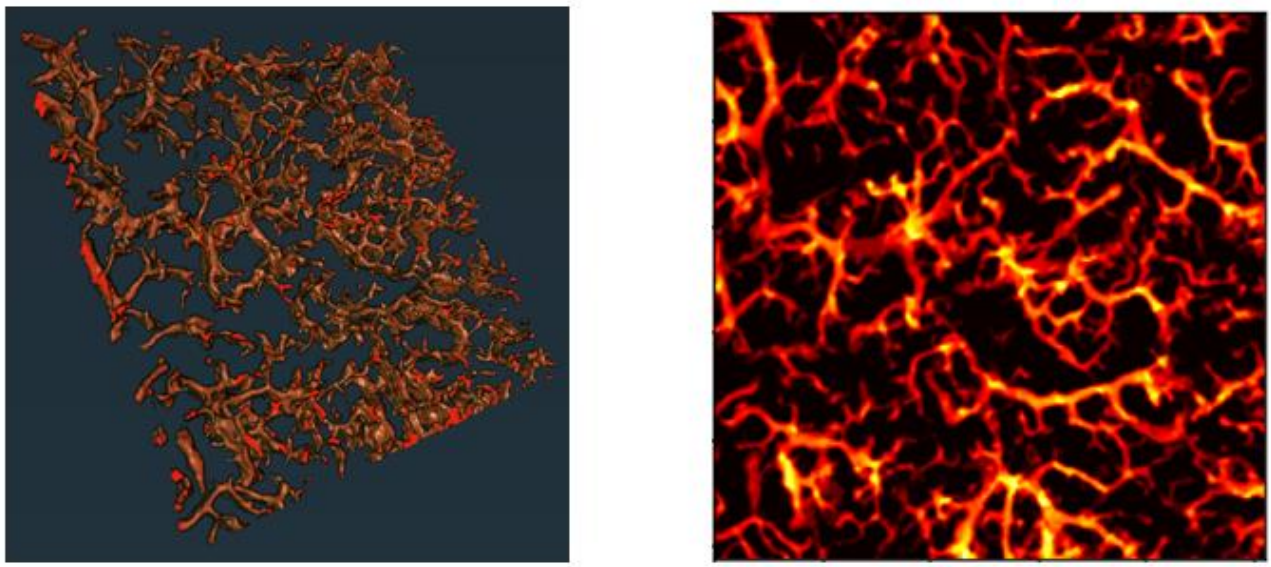


***Figure 3.6**: 3D view of semi-automatic segmentation obtained with Amira, and en face maximum intensity projection view.*

4. Deep Neural Network Architectures and Methods

4.1 DNN Architectures

In this work three different 3D Deep Neural Network (DNN) architectures have been used to segment 3D OCTA volumes. In particular, were tried some of the most used in medical image segmentation, which are the 3D version of the original 2D U-Net, that is the 3D U-Net, and two more versions of the 3D U-Net, namely the 3D Res-Unet, and the 3D Dense-Unet, which take their names from the backbones that have been used to replace the encoder part of the 3D U-Net. In the following sections the architectures of the networks and their implementation are described.

4.1.1 3D U-Net

The U-Net is a convolutional neural network developed for biomedical image segmentation in 2015 [23], that has been widely used in the past years, showing state-of-the-art results in many biomedical image segmentation tasks.
U-Net structure is particularly suited for the segmentation of images in an end-to-end setting, meaning that given a raw image as input, it is able to provide a segmentation as output.
The U-Net has been widely used for a variety of tasks, including segmentation of medical images, satellite images, and microscopy images. Its success can be attributed to its ability to handle small and large scale objects, its robustness to partial occlusions, and its ability to learn features at multiple scales.

Hence, given the necessity to segment vessels in OCTA volumes, the U-Net is well-suited for the task, since it is able to learn features at multiple scales, which is important for vessel segmentation as vessels can have different widths and thicknesses in OCTA volumes.

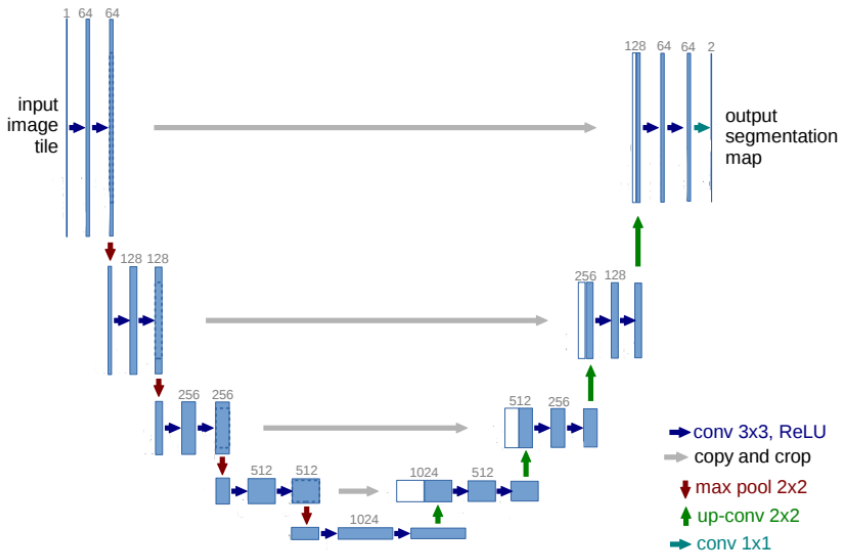


***Figure 4.1**: U-Net architecture. Each blue box corresponds to a multi-channel feature map, where the number of channels is denoted on top. The white boxes correspond to the copied and concatenated feature maps, and the arrows denote the different operations. Note that this is the original architecture taken from [23], while in our case the 2D operations are replaced by the corresponding 3D ones.*

The original U-Net has a U-shaped architecture (Figure 4.1) that consists of a contracting path, also called "encoder", and an expanding path, also called "decoder", which allows to capture contextual information from both local and global regions of the input image. The 3D U-Net used to segment the OCTA dataset previously described is an adaptation to 3D of the original 2D U-Net inspired by [24], in which the 2D operations are replaced by the corresponding 3D operations, such as 3D convolutions, 3D max-pooling and 3D up-convolutional layers. The architecture of the 3D U-Net is shown in (Figure 4.2).

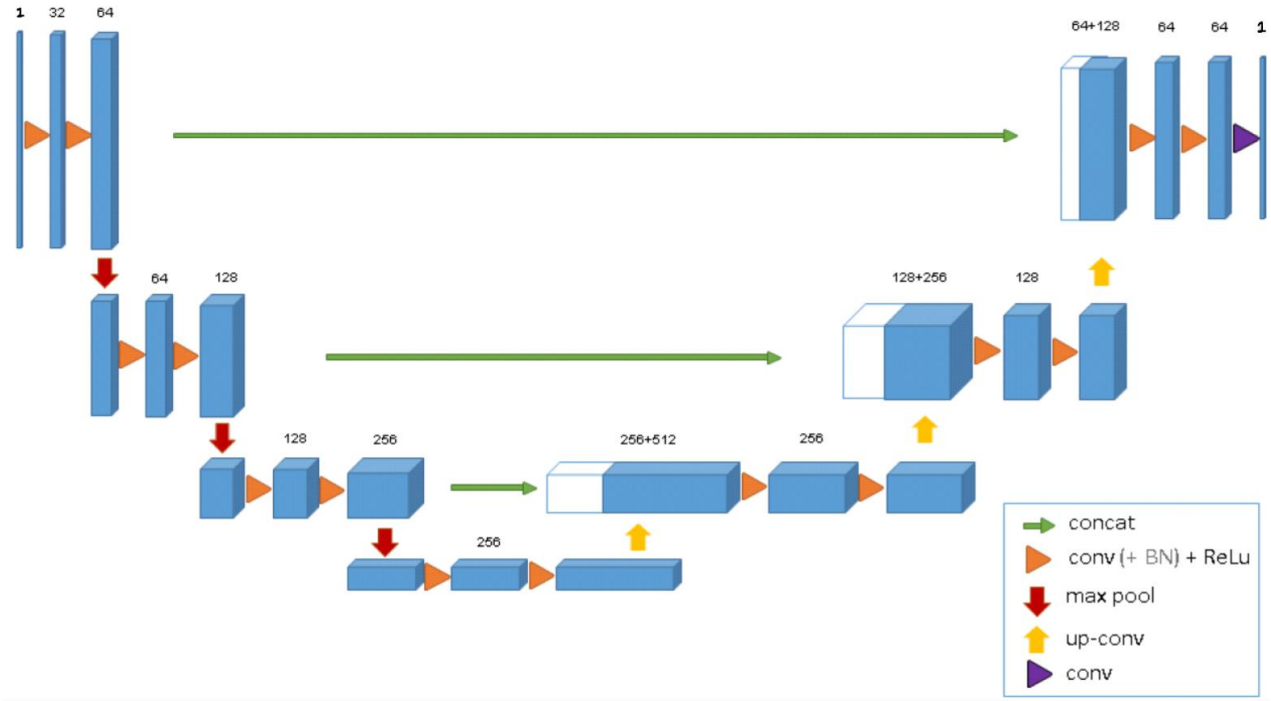


***Figure 4.2**: 3D U-Net architecture. Each blue box corresponds to a multi-channel feature map, where the number of channels is denoted on top. The white boxes correspond to the copied and concatenated feature maps, and the arrows denote the different operations. In this case there is only one channel for the input layer, since we are using grayscale volumes, and only one channel in the output layer, which correspond to the number of labels in our case (only vessels) [24].*

Like in the original architecture of the U-Net, the contracting path consists of a series of 3x3x3 convolutions, each followed by a rectified linear unit (ReLU) and a 2x2x2 max pooling layer that reduce the spatial resolution of the input image. The purpose of the contracting path is to extract features from the input image and reduce the dimensionality of the data. The ReLU function, which returns a value of zero for each negative input while keeping as they are positive inputs, is used to introduce non-linearity, making the model able to learn more complex types of data. The max pooling layers are used to down-sample the feature maps produced by the convolutional layers, by propagating the maximum activation from each 2x2x2 window into the next feature map. At each downsampling step the number of feature channels is doubled, while the size of the outputs is halved.

The expanding path, on the other hand, uses a series of upsampling and 2x2x2 convolutional layers to increase the spatial resolution of the feature maps produced by the contracting path. It also concatenates the feature map from the encoder path, and uses two 3x3x3 convolutions each followed by a ReLU like in the contracting path. The purpose of the expanding path is to recover the spatial resolution of the feature maps and produce a dense prediction for the input image. The upsampling layers are used to up-sample the feature maps, while the convolutional layers are used to refine the features.
The skip connections, concatenations, from the contracting path, allow the network to recover spatial resolution and incorporate context from earlier layers.

At the final layer a 1x1x1 convolution is used to map each 64-component feature vector to the desired number of classes, which is 1 in our case.

However, in order to segment the full 512x512x96 volumes, some changes were applied to the architecture described in Figure 4.5, by adding another layer, so that even using the full volumes, in the last layer there would be smaller resolutions to learn even small-scale features, and by changing the number of filters to (16,32,64,128,256), due to memory limitations to handle the higher number of filters with volumes this big.

The code to implement this modified 3D U-Net architecture is shown in Code 4.1.

```
def encoder_block(inputs, filters, kernel_size=(3,3,3), padding ='same',
pool_size=(2,2,2)):
    conv = tf.keras.layers.Conv3D(filters, kernel_size, padding=padding)(inputs)
    conv = tf.keras.layers.BatchNormalization()(conv)
    conv = tf.keras.layers.Activation('relu')(conv)
    conv = tf.keras.layers.Conv3D(filters*2, kernel_size, padding=padding)(conv)
    conv = tf.keras.layers.BatchNormalization()(conv)
    conv = tf.keras.layers.Activation('relu')(conv)
    pool = tf.keras.layers.MaxPooling3D(pool_size=pool_size)(conv)
    return pool, conv

def decoder_block(inputs, concat_tensor, filters, kernel_size=(3,3,3), padding
='same'):
    up = tf.keras.layers.UpSampling3D(size=2)(inputs)
    concat = tf.keras.layers.concatenate([concat_tensor, up], axis=-1)
    conv = tf.keras.layers.Conv3D(filters, kernel_size, padding=padding)(concat)
    conv = tf.keras.layers.BatchNormalization()(conv)
    conv = tf.keras.layers.Activation('relu')(conv)
    conv = tf.keras.layers.Conv3D(filters, kernel_size, padding=padding)(conv)
    conv = tf.keras.layers.BatchNormalization()(conv)
    conv = tf.keras.layers.Activation('relu')(conv)
    return conv

def bottleneck(inputs, filters, kernel_size=(3,3,3), padding ='same'):
    conv = tf.keras.layers.Conv3D(filters, kernel_size, padding=padding)(inputs)
    conv = tf.keras.layers.BatchNormalization()(conv)
    conv = tf.keras.layers.Activation('relu')(conv)
    conv = tf.keras.layers.Conv3D(filters*2, kernel_size, padding=padding)(conv)
    conv = tf.keras.layers.BatchNormalization()(conv)
    conv = tf.keras.layers.Activation('relu')(conv)
    return conv

# define input shape
input_shape = (512, 512, 96, 1)

# define input and output
inputs = tf.keras.layers.Input(shape=input_shape)

# define encoder
pool1, conv1 = encoder_block(inputs, 8)
pool2, conv2 = encoder_block(pool1, 16)
pool3, conv3 = encoder_block(pool2, 32)
pool4, conv4 = encoder_block(pool3, 64)

bn = bottleneck(pool4, 128)

# define decoder
dec4 = decoder_block(bn, conv4, 128)
dec3 = decoder_block(dec4, conv3, 64)
dec2 = decoder_block(dec3, conv2, 32)
dec1 = decoder_block(dec2, conv1, 16)

outputs = tf.keras.layers.Conv3D(1, (1, 1, 1), activation='sigmoid')(dec1)

model = tf.keras.Model(inputs, outputs)
```

***Code 4.1**: Implementation code of 3 U-Net that recreates the architecture to segment 512x512x96 volumes.*

A summary of the model and the layers is shown in Figure 4.3.

```
__________________________________________________________________________________________
 Layer (type)                 Output Shape      Param #     Connected to
==========================================================================================
input_1                 [(None,512,512,96,1)]    0             []
conv3d                   (None,512,512,96,8)     224     ['input_1[0][0]']
batch_normalization   (None,512,512,96,8)       32       ['conv3d[0][0]']
activation              (None,512,512,96,8)      0    ['batch_normalization[0][0]']
conv3d_1               (None,512,512,96,16)      3472    ['activation[0][0]']
batch_normalization_1   (None, 512,512,96,16)   64       ['conv3d_1[0][0]']
activation_1            (None,512,512,96,16)     0  ['batch_normalization_1[0][0]']
max_pooling3d          (None,256,256,48,16)      0       ['activation_1[0][0]']
conv3d_2                (None,256,256,48,16)     6928      ['max_pooling3d[0][0]']
batch_normalization_2   (None,256,256,48,16)     64         ['conv3d_2[0][0]']
activation_2         (None,256,256,48,16)        0  ['batch_normalization_2[0][0]']
conv3d_3             (None,256,256,48,32)        13856    ['activation_2[0][0]']
batch_normalization_3   (None,256,256,48,32)     128        ['conv3d_3[0][0]']
activation_3         (None,256,256,48,32)        0  ['batch_normalization_3[0][0]']
max_pooling3d_1    (None,128,128,24,32)          0        ['activation_3[0][0]']
conv3d_4               (None,128,128,24,32)      27680   ['max_pooling3d_1[0][0]']
batch_normalization_4   (None,128,128,24,32)     128        ['conv3d_4[0][0]']
activation_4         (None,128,128,24,32)        0  ['batch_normalization_4[0][0]']
conv3d_5              (None,128,128,24,64)       55360       ['activation_4[0][0]']
batch_normalization_5   (None,128,128,24,64)     256        ['conv3d_5[0][0]']
activation_5         (None,128,128,24,64)        0  ['batch_normalization_5[0][0]']
max_pooling3d_2       (None,64,64,12,64)         0          ['activation_5[0][0]']
conv3d_6                (None,64,64,12,64)       110656   ['max_pooling3d_2[0][0]']
batch_normalization_6 (None,64,64,12,64)        256      ['conv3d_6[0][0]']
activation_6         (None,64,64,12,64)          0  ['batch_normalization_6[0][0]']
conv3d_7             (None,64,64,12,128)         221312      ['activation_6[0][0]']
batch_normalization_7   (None,64,64,12,128       512        ['conv3d_7[0][0]']
activation_7         (None,64,64,12,128)         0  ['batch_normalization_7[0][0]']
```

```
max_pooling3d_3    (None,32,32,6,128           0         ['activation_7[0][0]']

conv3d_8               (None,32,32,6,128)      442496    ['max_pooling3d_3[0][0]']

batch_normalization_8    (None,32,32,6,128)  512         ['conv3d_8[0][0]']

activation_8        (None,32,32,6,128)        0     ['batch_normalization_8[0][0]']

conv3d_9              (None,32,32,6, 256)     884992    ['activation_8[0][0]']

batch_normalization_9  (None,32,32,6,256)    1024        ['conv3d_9[0][0]']

activation_9        (None,32,32,6,256)        0     ['batch_normalization_9[0][0]']

up_sampling3d     (None,64,64,12,256)         0           ['activation_9[0][0]']

concatenate         (None,64,64,12,384)       0         ['activation_7[0][0]',
                                                         'up_sampling3d[0][0]']
conv3d_10               (None,64,64,12,128)  1327232      ['concatenate[0][0]']

batch_normalization_10  (None,64,64,12,128) 512         ['conv3d_10[0][0]']

activation_10       (None,64,64,12,128)       0    ['batch_normalization_10[0][0]']

conv3d_11            (None,64,64,12,128)      442496       ['activation_10[0][0]']

batch_normalization_11 (None,64,64,12,128)   512         ['conv3d_11[0][0]']

activation_11       (None,64,64,12,128)       0    ['batch_normalization_11[0][0]']

up_sampling3d_1    (None,128,128,24,128)      0           ['activation_11[0][0]']

concatenate_1      (None,128,128,24,192)      0         ['activation_5[0][0]',
                                                         'up_sampling3d_1[0][0]']
conv3d_12           (None,128,128,24,64)      331840    ['concatenate_1[0][0]']

batch_normalization_12 (None,128,128,24,64) 256         ['conv3d_12[0][0]']

activation_12       (None,128,128,24,64)      0    ['batch_normalization_12[0][0]']

conv3d_13            (None,128,128,24,64)     110656       ['activation_12[0][0]']

batch_normalization_13 (None,128,128,24,64) 256         ['conv3d_13[0][0]']

activation_13       (None,128,128,24,64)      0    ['batch_normalization_13[0][0]']

up_sampling3d_2    (None,256,256,48,64)       0          ['activation_13[0][0]']

concatenate_2      (None,256,256,48,96)       0        ['activation_3[0][0]',
                                                         'up_sampling3d_2[0][0]']
conv3d_14           (None,256,256,48,32)      82976     ['concatenate_2[0][0]']

batch_normalization_14 (None,256,256,48,32) 128         ['conv3d_14[0][0]']

activation_14       (None,256,256,48,32)      0    ['batch_normalization_14[0][0]']

conv3d_15            (None,256,256,48,32)     27680       ['activation_14[0][0]']

batch_normalization_15 (None,256,256,48,32) 128         ['conv3d_15[0][0]']

activation_15       (None,256,256,48,32)      0    ['batch_normalization_15[0][0]']
```

```
up_sampling3d_3    (None,512,512,96,32)      0        ['activation_15[0][0]']

concatenate_3      (None,512,512,96,48)      0         ['activation_1[0][0]',
                                                        'up_sampling3d_3[0][0]']

conv3d_16           (None,512,512,96,16)     20752      ['concatenate_3[0][0]']

batch_normalization_16 (None,512,512,96,16) 64          ['conv3d_16[0][0]']

activation_16       (None,512,512,96,16)     0    ['batch_normalization_16[0][0]']

conv3d_17            (None,512,512,96,16)    6928       ['activation_16[0][0]']

batch_normalization_17 (None,512,512,96,16) 64          ['conv3d_17[0][0]']

activation_17       (None,512,512,96,16)     0    ['batch_normalization_17[0][0]']

conv3d_18               (None,512,512,96,1)  17           ['activation_17[0][0]']

==================================================================================
==================
Total params: 4,122,449
Trainable params: 4,120,001
Non-trainable params: 2,448
```

***Figure 4.3**: Summary of the 3D U-Net implemented, where all the layers and shapes in each layer can be seen.*

Additionally, also the original architecture described in [24] was tested, to check whether the higher number of filters could help the model learn more complex features from the volumes and improve the performances. This model was trained on subvolumes of shape 128x128x96 pixels, since the whole volumes could not be handled by this architecture with higher number of filters. The code to implement this architecture of the 3D U-Net is the same shown in Code 4.1, with few modifications.

This architecture has 16 million parameters, four times as many as the architecture used to segment the whole volumes, due to the higher number of filters.

4.1.2 3D Res-Unet

ResNet, short for Residual Network, is a deep learning architecture that was introduced in 2015 [25] and has been widely used since then in many and various tasks, semantic segmentation included.

It has been shown that increasing the depth of a network can increase the performances [25,26], but many problems usually arise going deeper. One of the problems was the vanishing gradient, which refers to the phenomenon that occurs when the gradients of the weights with respect to the loss become very small as they are backpropagated through many layers, almost "vanishing", hence not being able to effectively update the weights. Although this problem has already been overcome using normalized initializations, and intermediate normalization layers, another problem of degradation of performance arises when going deeper in a network. They solved this problem in the ResNet by introducing the use of "residual connections", which are simple shortcut connections that performs identity mapping, and connect their outputs to the outputs of the stacked layers (Figure 4.4) via element-wise addition, without increasing computational complexity.

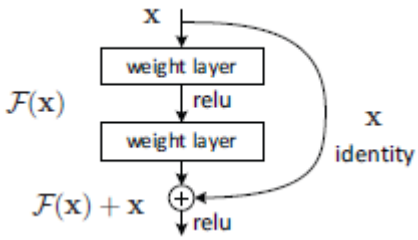


***Figure 4.4**: an example of residual connection block [25].*

These blocks allow the gradients to flow more easily through the network even in very deep neural networks, solving the vanishing gradient problem in the initial layers. With the use of these blocks, even by going very deep in the network (152 layers in the ResNet-152), they have shown significant accuracy gains, by winning many international competitions on challenging datasets. Another positive side of this architecture is that despite being deeper than its predecessors, for example the VGG-16/19 nets, it still has much lower complexity, even comparing the very deep ResNet-152 (around 11 billion FLOPs, against 15/19).

The architecture for the ResNet-34 is shown in Figure 4.5, and on top of this architecture is built the same decoder path described previously, without the last layer of the ResNet, to recreate the U-shape architecture of the 3D U-Net.

To implement the 3D Res-Unet the "*segmentation_models_3D*" library in Python has been used, from [27,28], that allows the simple implementation of 3D U-Net with many backbones.

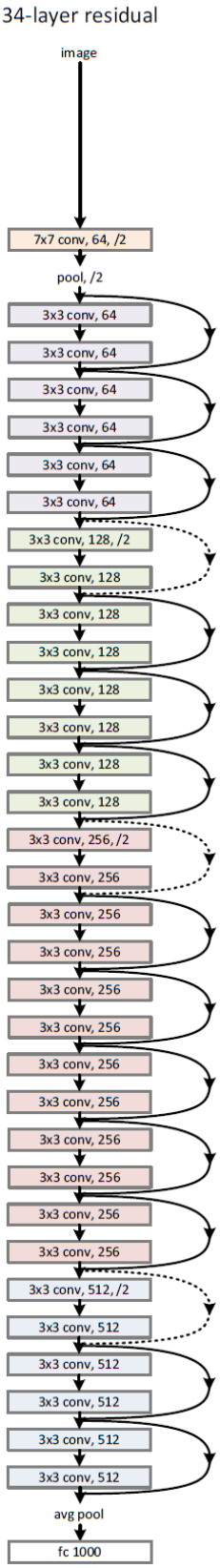


***Figure 4.5**: ResNet-34 architecture with residual connection (skipped dots line indicates increase in dimension, obtained extra zero entries padded) [25]. Downsampling in this case is done with convolutional layers with stride of 2.*

4.1.3 3D Dense-Unet

DenseNet [30] is a CNN published the year after ResNet, with which it shares some similarities. The fundamental block in the DenseNet is a dense block that follows a similar idea of connectivity between layers adopted in the residual block, but at the same time yields very different behaviors.

The dense block, fundamental structure of the DenseNet, contains multiple layers, each with batch normalization, a rectified linear unit (ReLU) activation function, and a 3x3 convolutional layer, and each layer is connected to all subsequent layers, as shown in Figure 4.6. Therefore, each layer receives inputs from all the preceding layers, and in the same way, its output feature maps will be passed to all subsequent layers.

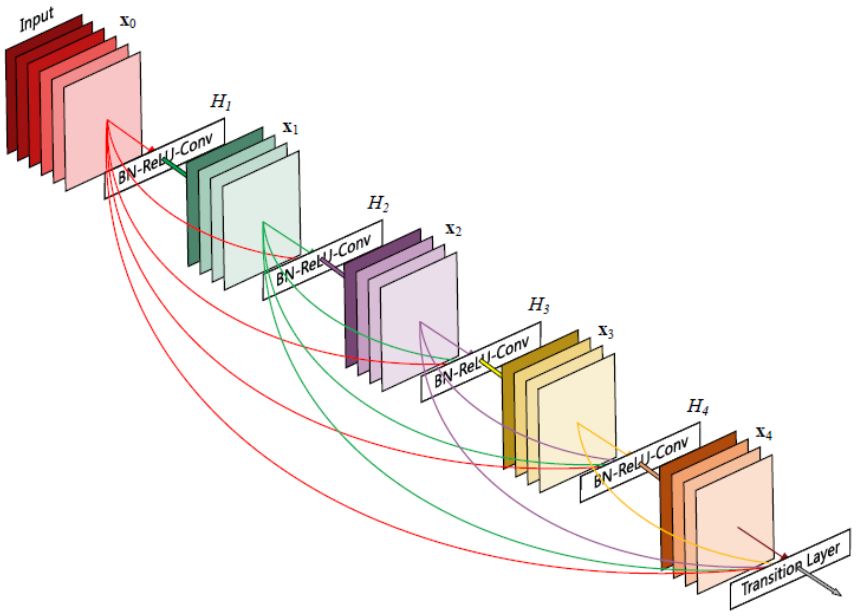


***Figure 4.6**: Example of dense block with 5 layers where the dense connectivity between all the layers can be observed.*

This dense connectivity ensures that all the possible information is passed between layers in the network, since each layer also passes information that needs to be preserved throughout the network. With this connectivity, the problem of the vanishing gradient is also addressed efficiently.

The main difference between the connection in ResNets and DenseNets is that the inputs in the residual blocks are summed, while in the dense block they are concatenated. The concatenation of the inputs allows the layers in the network to access the feature maps learned by any of the previous layers, encouraging the feature reuse, which leads to a more compact model and better parameter efficiency and lower computational complexity than state-of-the-art methods, as shown in the original paper [30].

There are different variants of DenseNet that have been proposed which differ in the total number of layers present in the architecture, and for our task we used the DenseNet-169, where the number indicates the number of total layers in the architecture, which is shown highlighted in Figure 4.7.

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48	[1 × 1 conv, 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

***Figure 4.7**: DenseNet architecture, which consists of 4 dense blocks with variable number of layers. Each conv layer corresponds to Batch normalization-ReLU-Conv [30].*

As for the 3D ResUnet, the decoder part is built on top of this architecture, and the network is implemented using the same library mentioned earlier.

4.2 Dual-channel input

Although it is not another architecture, another approach to improve the performances of the previously described architectures was tested, which consists of utilizing a dual-channel input to the DNNs, where the second channel is a Meijering [31] filtered volume derived from the original data, which is passed in the first channel.

As demonstrated in [32], the Meijering filter is a good vesselness filter which can help improve the performances of segmentation with respect to the original data.

The Meijering filter is a morphological filter that has the ability to emphasize specific features in the data, such as edges or lines, which can be crucial for the segmentation task. By incorporating the filtered volume as a second channel in the input to the DNNs, the network is able to make use of both the original and filtered data in the segmentation process. This might enhance the performance of the segmentation by providing additional information to the network and mitigating the effect of noise or variability in the original data.

The Meijering filter is a Hessian-based vessel enhancement filter, based on the second order derivatives of the image intensity, which identify the curvilinear structures in the images. The Hessian matrix of an image f(x,y,z) is defined as:

$$H(f) = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_1 \partial x_3} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \frac{\partial^2 f}{\partial x_2 \partial x_3} \\ \frac{\partial^2 f}{\partial x_3 \partial x_1} & \frac{\partial^2 f}{\partial x_3 \partial x_2} & \frac{\partial^2 f}{\partial x_3^2} \end{bmatrix} \quad \textbf{(1)}$$

A standard deviation σ Gaussian kernel is applied to the image together with the second derivative to reduce the noise and tune the filter to the width of the structures, making the filter a multi-scale framework where the vessel scale depends on the σ [32].

The direction of the vessel is identified by the first eigenvectors of H(f), e_1, which represents the direction along which the second order derivative is maximum.

The Meijering filter is based on a modified Hessian matrix, defined as:

$$H'(f) = \begin{bmatrix} h_{11} + \frac{\alpha}{2}(h_{22} + h_{33}) & (1 - \frac{\alpha}{2})h_{12} & (1 - \frac{\alpha}{2})h_{13} \\ (1 - \frac{\alpha}{2})h_{21} & h_{22} + \frac{\alpha}{2}(h_{11} + h_{33}) & (1 - \frac{\alpha}{2})h_{23} \\ (1 - \frac{\alpha}{2})h_{31} & (1 - \frac{\alpha}{2})h_{32} & h_{33} + \frac{\alpha}{2}(h_{11} + h_{33}) \end{bmatrix} \quad \textbf{(2)}$$

Its eigenvalues with respect to the Hessian matrix H(f) are defined as

$$\lambda_i' = \lambda_i + \alpha\lambda_j + \alpha\lambda_k \quad \textbf{(3)}$$

Where α=1/3 usually.

The vesselness is then defined by

$$F = \begin{cases} \lambda_{max}/\lambda_{min} & \lambda_{max} < 0 \\ 0 & \lambda_{max} \geq 0 \end{cases} \quad \textbf{(4)}$$

where, λ'_{max}= max $\{\lambda'_1, \lambda'_2, \lambda'_3\}$, which is computed at each voxel and λ_{min} is the minimum of all $\lambda'_{max}$of the image.

This filter was applied to all the 214 volumes in the dataset, and the volumes were saved as dual-channel volumes, and then used as input to one of the DNNs described in the previous sections.

The simplified architecture will be like shown in Figure 4.8.

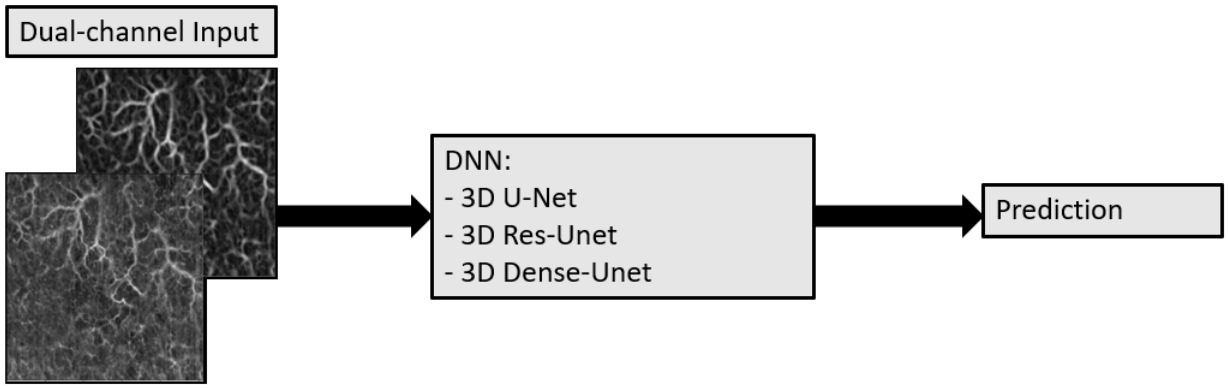


Figure 4.8. *Simplified scheme of this proposed DNN approach, in which the inputs the three DNNs architecture are the original and Meijering filtered volume.*

4.3 Ensemble Learning

Ensemble Learning is a powerful tool for improving model performance by combining the outputs of multiple models, each of which may capture different aspects of the data and have different strengths and weaknesses. The goal of Ensemble Learning is to reduce the variance and overfitting of individual models, while increasing the overall accuracy and robustness of the predictions.

4.3.1 Majority Voting

There are many ways to apply Ensemble Learning, among which the ones that was implemented, called "*majority voting*", which is one of the simpler Ensemble Learning methods.

In binary segmentation tasks, each pixel can be assigned to one of two classes (in our case either vessel or background), and majority voting simply counts the number of models that predict class 0 and the number of models that predict class 1 for each pixel. Then, the class with the highest count is assigned to that pixel. This process is repeated for all pixels in each input sample, and the final prediction for each sample is based on the majority votes across all pixels.

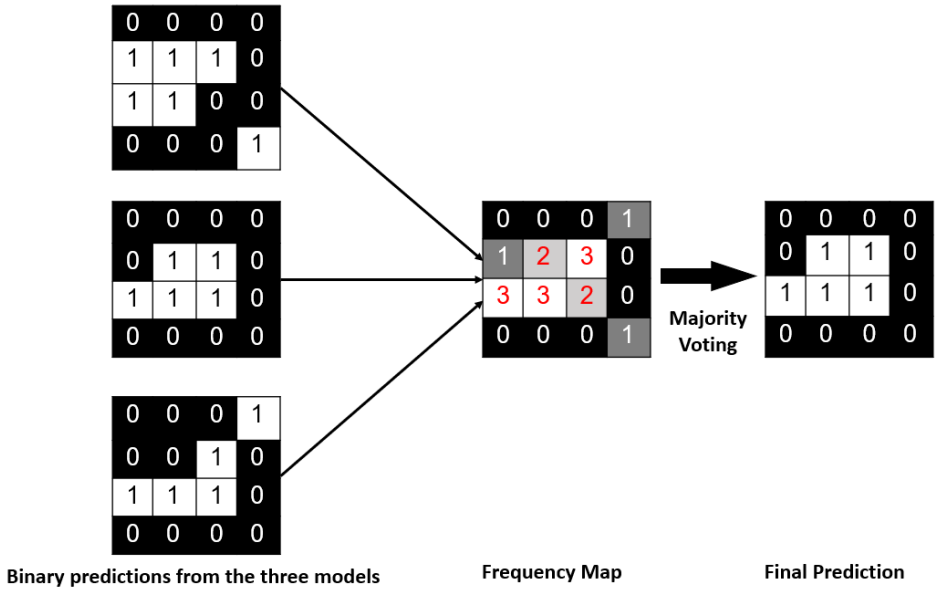


Figure 4.9. *Example of how the predictions of different models are merged through majority voting. After combining the individual predictions in a frequency map only the pixels that are present in the majority of the predicted volumes (at least two) remain in the final prediction.*

One advantage of majority voting is its simplicity and ease of implementation. Compared to other Ensemble Learning methods, such as bagging or boosting, majority voting does not require any additional training or adjustment of model parameters. This makes it an attractive option for practical applications where the time and resources for training multiple models may be limited. Additionally, majority voting can help to mitigate the effects of outliers or noise in the data, since the majority vote tends to be more robust to these effects than the individual model predictions. This could be useful with our dataset, where noise and artifacts are present in some samples.

Overall, majority voting is a useful technique for combining the outputs of multiple models in Ensemble Learning, which can lead to better performances than the individual models, without requiring any additional training time.

4.3.2 Dynamic Model Selection

Another possible Ensemble Learning method is the Dynamic Model Selection, a technique that refers to the process of selecting the appropriate model architecture from a set of candidate models during the training or inference stage, based on the characteristics of the input data and the performance of the models on that data.

In this work, the implementation of Dynamic Model Selection involves the training of multiple models with different architectures, the three model architectures previously described, and the combination of their predictions during the inference stage to improve overall performances.

The model architecture for each prediction is hence selected based on some criteria, which in this case was the prediction with the highest Dice Score.

Therefore, the final predictions will be comprised of predictions from the three different model architectures, based on which gives highest Dice performances in each specific sample.

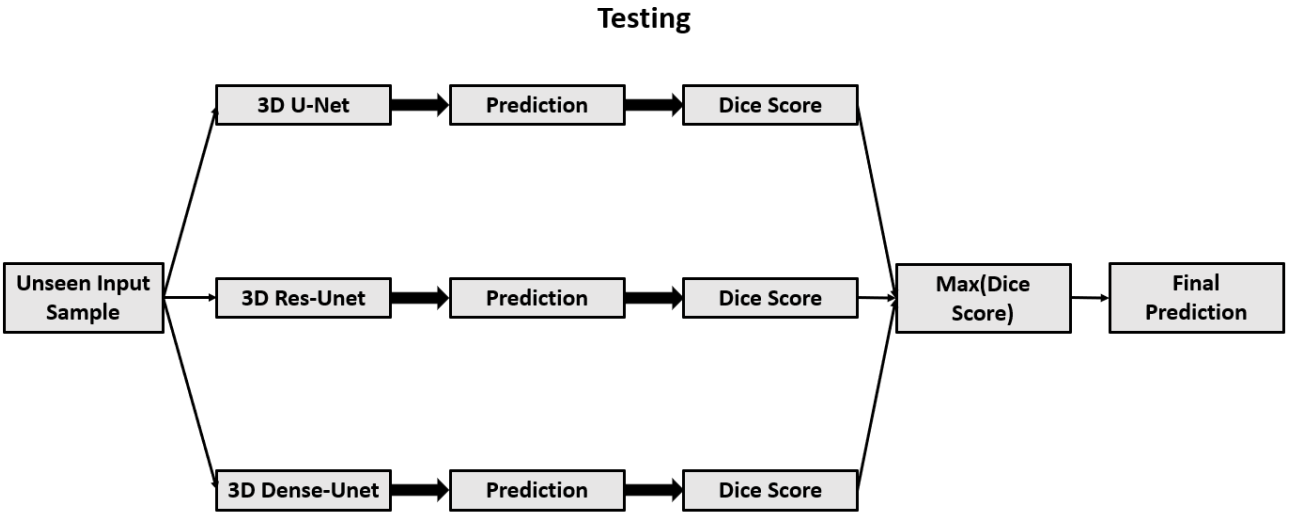


Figure 4.10. *Schematic diagram of the Dynamic Model Selection process: each sample in the Test Set is segmented by each of the three trained model architectures, then the Dice Score is computed for each prediction, and the prediction with the highest Dice Score between the three architectures is chosen as final prediction for that sample.*

5 Evaluation Metrics and Losses

5.1 Evaluation Metrics

Since the task of this thesis work is image segmentation, in particular binary segmentation, meaning the goal is trying to classify each pixel as vessel or background, some type of metrics are needed to evaluate the neural network performances in recognizing whether a pixel belongs or not to blood vessels, when comparing the prediction to the ground truth.

This step is important in order to check if the chosen hyperparameters or even the architecture of the neural network are good for the specific task.

It is also important to choose the right metrics, since not every metric is always useful in binary segmentation, especially in a case where high unbalance between background and vessel pixels is present, like in our dataset where the pixels belonging to the vessel class are the 15% of the total number of pixels.

For example, the classic pixel-wise accuracy would not be representative of the neural network performances in this case, given the fact that background pixels are the significant majority, and even not recognizing vessels pixels belonging to the minority class would still yield high accuracy values.

The metrics used to evaluate the neural network performances in this study are the *Intersection over Union (IoU) score***,** also known as Jaccard index, the *Dice coefficient (*or *F1-score), precision* and *recall* which are among the most common evaluation metrics used in image segmentation.

- *Precision*: Precision is the number of true positive pixels TP (pixels belonging to vessel class predicted as belonging to vessel class) divided by the total number of true positive TP and false positive FP (pixels belonging to background predicted as belonging to vessel class) pixels (1).
  In other words, it measures the proportion of correctly identified object pixels out of all the pixels that were identified as object pixels.

$$Precision = \frac{TP}{TP+FP} \qquad (1)$$

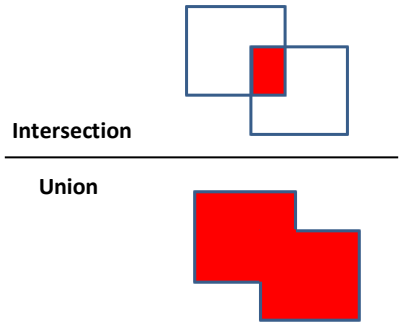
- *Recall*: Recall is the number of true positive TP pixels divided by the total number of true positive TP and false negative FN pixels (pixels belonging to vessel class predicted as background) (2). It is a measure of how well the model identifies instances of the object of interest.

$$Recall = \frac{TP}{TP+FN} \qquad (2)$$

- *IoU-score*: The Intersection over Union score, also known as Jaccard similarity is a good metric for binary segmentation algorithms evaluation because it is easy to interpret, and since it is widely used, it makes it easier to compare results with other studies. The IoU score measures the similarity between two sets of data, the predicted segmentation and the ground truth segmentation. The score is calculated as the ratio of the intersection of the predicted segmentation and the ground truth segmentation to the union of the two sets (3). An IoU score of 1 indicates a perfect match between the predicted and ground truth segmentations, while a score of 0 indicates no overlap

$$IoU_{score} = \frac{TP}{TP + FP + FN}$$

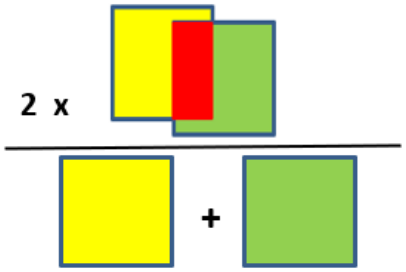
$$IoU = \frac{Area\ of\ Intersection}{Area\ of\ Union} \quad (3)$$



- *Dice coefficient*: It is calculated as two times the number of common elements (true positives) divided by the number of elements in each set, or from a graphical point of view, two times the intersection divided by the sum between union and intersection (4). It takes into account both precision and recall. Dice coefficient ranges from 0 to 1, where 1 indicates a perfect overlap between the predicted and ground truth segmentation, and 0 indicates no overlap.

$$F1_{score} = \frac{2(Precision * Recall)}{Precision + Recall}$$

$$Dice_{score} = \frac{2 * TP}{2 * TP + FP + FN} \quad (4)$$



Additionally, we also implemented another metric which gives a general idea of how continuous or disconnected the predicted segmentation is, which is very important in the case of vascular network, even more than obtaining high performances on the volumetric metrics.

This metric, that we called "*Disconnectivity*", computes for each sample volume the ratio between the number of "fragments", meaning disconnected components, in the predicted segmentation and the number of fragments in the ground truth mask. The lower the value, the more connected and continuous the segmentation is. To identify different components, we used a connectivity of 26, where voxels are considered connected to a central voxel, if they are in a cube, as shown in Figure 5.1.

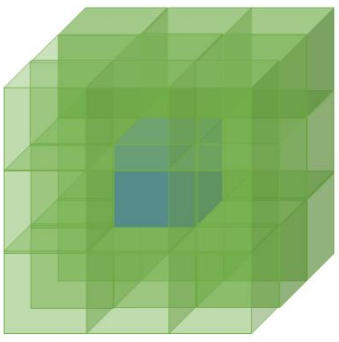


***Figure 5.1**. 26 connectivity in 3D.*

5.2 Loss Functions

A loss function is a mathematical function used to evaluate the performance of a model that calculates the error (loss) between the predicted output and the actual output for a given set of inputs. The goal of training a deep learning model is to find the set of parameters that minimize the value of the loss function. Once the loss function is defined, an optimization algorithm is used to minimize the value of the loss function, and it accomplishes this by "backpropagating" the error of the network, updating the weights of the network by multiplying the derivative of the loss function with respect to the derivative of each weight for the learning rate and subtracting this value from the weights of the network in the previous steps. Basically, the model is learning which value to assign to each weight based on how these changes are affecting the loss function.

This process is repeated for a certain number of iterations or until a stopping criterion is met.

In order to minimize the loss functions, we used the Adam (Adaptive Moment Estimation) optimizer, one of the most used optimizers, that is computationally efficient and well suited for large datasets due to its convergence properties and stability. The learning rates are updated for each iteration and depend on the mean and variance of the gradients observed so far.

During our work three different loss functions have been used.

- *Dice loss*: It is calculated as:

$$Dice_{loss} = 1 - Dice_{score} \qquad (5)$$

  it is less sensitive to class imbalance compared to other loss functions such as binary cross-entropy since it is calculated based on the intersection of the predicted and true masks, rather than their individual values. Although, it might not be the best choice, because of the intrinsically instability of its gradient, that becomes most evident with highly class imbalanced data where small denominators are present in the gradient calculations [33].

- *Jaccard loss*: it is obtained from the IoU-score, but we decided to implement it using the negative logarithm of the IoU-score, as proposed in [34, 35], so that the loss is more stable during training.

$$Jaccard_{loss} = -\log\left(IoU_{score}\right) \quad (6)$$

- *CenterlineDice (clDice) loss:* as described in [36], when dealing with segmentation of vascular structures, traditional volumetric metrics such as Dice and IoU can be sub-optimal to evaluate a model performance. That is why the authors in [36] introduced a new metric, and loss, that aims to preserve topology of the vascular network, while also yielding accurate segmentations.
  The proposed loss combines the Dice Loss and a new loss, the centerline Dice loss, in which the ground truth and the predicted mask are first "soft-skeletonized", and then precision between predicted skeleton (S_P) and ground truth mask (V_L), and recall between ground truth skeleton (S_L) and predicted mask (V_P) are combined as an harmonic mean to compute the clDice. The formula is shown in Eq. (7).

$$clDice(V_P, V_L) = 2\ x\ \frac{T_{prec}(S_P, V_L)\ x\ T_{sens}(S_L, V_P)}{T_{prec}(S_P, V_L) + T_{sens}(S_L, V_P)} \quad (7)$$

The soft-skeleton is obtained through an iterative process of max- and min-pooling in order to create a skeletonization that can be differentiated.

The loss function can then be written as:

$$clDice_{loss} = (1 - \alpha)(1 - Dice) + \alpha(1 - clDice) \quad (8)$$

where alpha=0.5 to try to obtain a model that can give high performances on both volumetric metrics and topology metrics.

6. Results

In this chapter, we present a comprehensive evaluation of the performance of the deep neural network architectures previously described for the task of 3D OCTA volumes segmentation. We conducted extensive experiments using a dataset of 214 volumes acquired using our custom OCT system described in Section 3.1. The 214 volumes have been randomly split between training set, validation set and test set, using a ratio of 50-25-25%, 107-53-54 volumes respectively. We choose this partition based on the consideration that given the size of the volumes and the repetitiveness of the features, there would not be a need for a larger training set, and we preferred to have more samples on the test set, in order to better understand if the model is able to generalize the unseen data. The models will be therefore evaluated on the test set, which is comprised of 54 sample volumes that have not be seen during the training by the model.

To evaluate the performance of the different architectures, we used Dice Score, IoU score, Precision, Recall and the Disconnectivity metric. Our experiments were conducted using different hyperparameters and loss functions to find the optimal configuration for each network.

We compare the performance of each architecture in terms of the evaluation metrics used and highlight the strengths and weaknesses of each approach. The results provide valuable insights into the impact of different hyperparameters and loss functions on the performance of each network.

We also trained the models applying data augmentation to our training set.
The idea behind data augmentation is to artificially increase the size of the training dataset by creating additional training examples from the existing data. This is achieved by applying a set of pre-defined transformations, such as random rotation and flipping in our case, to the original volumes. These two transformations have been used because they do not affect the structure of the vascular network. The resulting augmented volumes are then used in training, to improve the robustness of the model and prevent overfitting.

Since acquiring large amounts of annotated 3D OCTA skin volume data can be time-consuming and expensive, data augmentation can be a valuable tool for increasing the size of the training dataset without the need for additional data. Additionally, data augmentation can help to improve the generalization performance of the model by making it less sensitive to variations in the dataset, such as changes in orientation, scale, or intensity. In Figure 6.1 examples of augmented samples are shown.

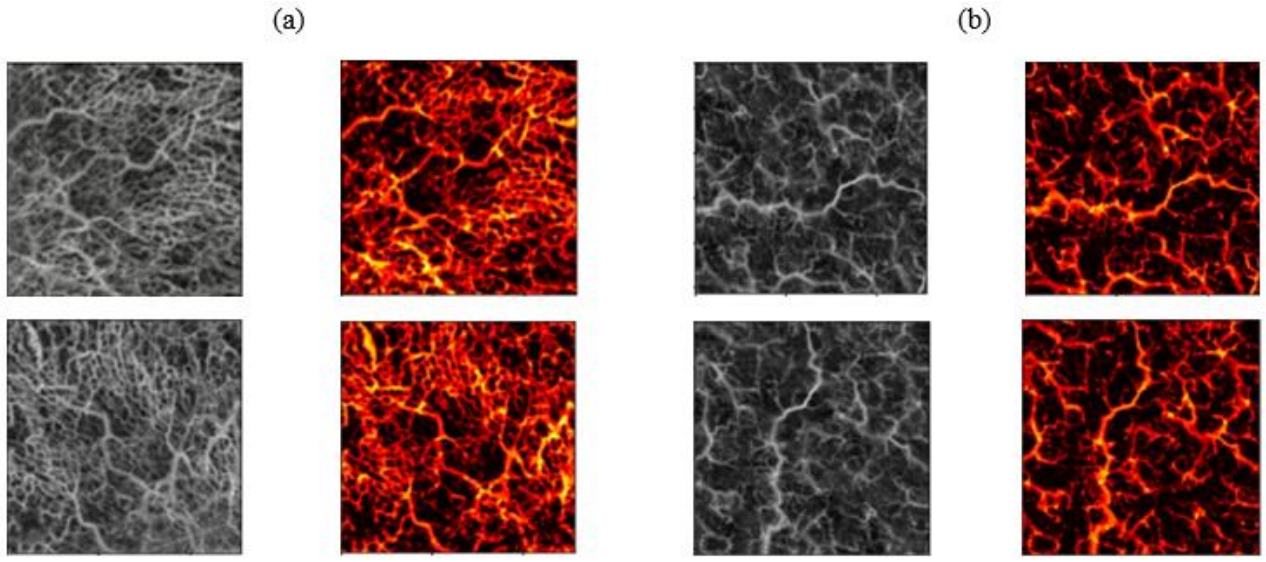


Figure 6.1. *Two examples of data augmentation on the data. Top row – en face view of MIP original samples and ground truth. Bottom row – augmented samples and ground truths. The augmentation is obtained through random rotation and flips.*

A simple schematic diagram that illustrates the procedure followed in this work is illustrated in Figure 6.2.

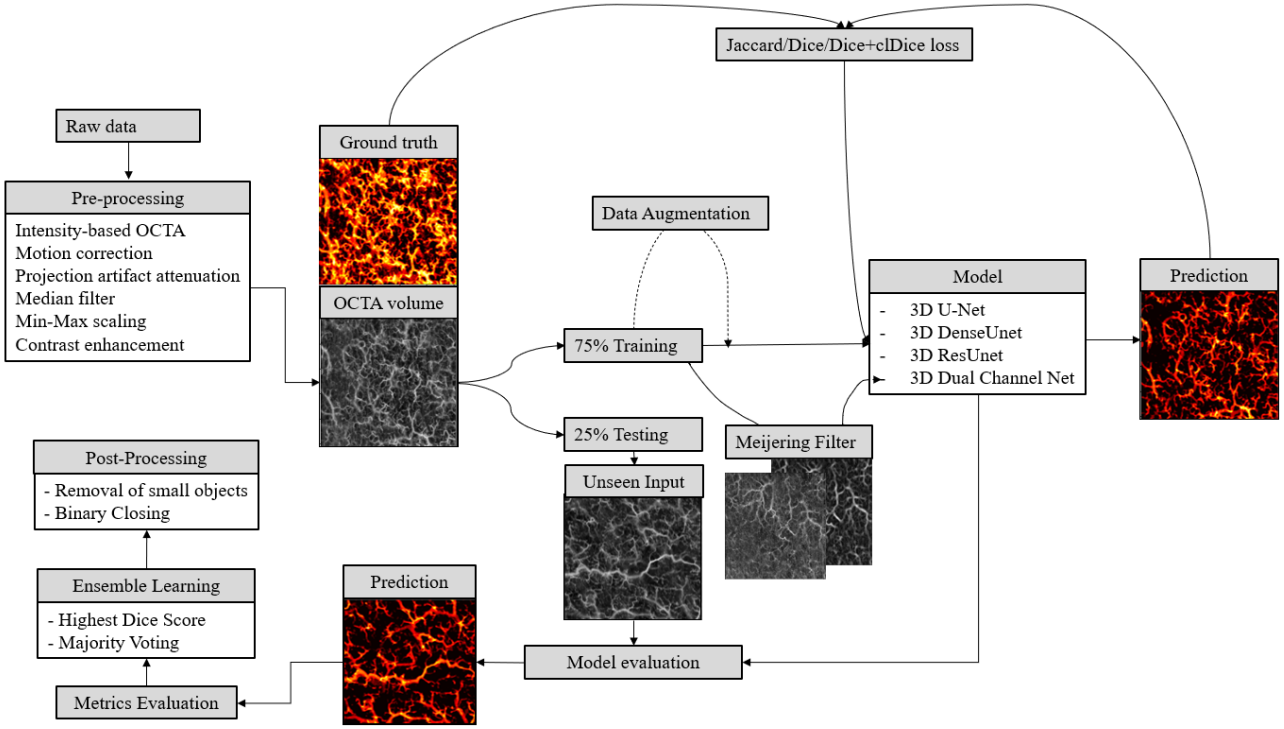


Figure 6.2. *Schematic diagram that illustrates the main steps followed in our work: pre-processing of raw data acquired to obtain the OCTA volume, segmentation of the volume using Amira to obtain the ground truth, split of the dataset in Training (and Validation) Set and Test Set, training of the different DNNs (using also data augmentation) with the three different loss functions that updates the parameters of the model during training, testing of the model on new unseen sample volumes to evaluate the performances. Note that we show en face view of the MIP of the volumes, but the models take in input and give in output 3D volumes.*

All the algorithms were implemented using the Tensorflow framework and Keras. The models were trained and tested on one NVIDIA RTX A6000, which allows us to handle the whole volumes despite their size.

6.1 3D U-Net results

In this section we present the results in terms of metrics, obtained with the 3D U-Net architecture described in Section 4.1. In particular, we show and compare the results obtained using the three different loss functions described in Section 5.2. Finally, the best performing configuration has been also tested applying data augmentation, to check whether the consideration done on the amount of data in the training set was correct.

Finally, we decided also to train the best performing configuration on the same dataset, in which every 512x512x96 volume has been split in 16 subvolumes of shape 128x128x96. This way we were able to test also the original 3D U-Net architecture proposed in [24], and check if the higher number of filters could allow to learn more complex features and give better performances, or if our architecture with a lower number of filters is enough.

To train the 3D U-Net we set a patience of 10 epochs with EarlyStopping, meaning that if the loss on the Validation Set computed during training does not improve for 10 epochs the training stops because it reached a plateau. The maximum number of epochs was set to 100, although it was never reached by any architecture. The learning rate was set to 10^{-4} for every architecture, which is a common starting value for the learning rate, and in any case the Adam optimizer changes it during training, being less sensitive to the initial value. The batch size was set to 1, since more than one volume of 512x512x96 pixels could not fit in the memory at the same time.

In Figure 6.3 the training and validation loss curves are shown, while Table 6.1 shows the evaluation metrics described in Section 5.1 on the original Test Set (54 volumes of shape 512x512x96). The metrics shown are computed on the minority class, meaning on the vessels.

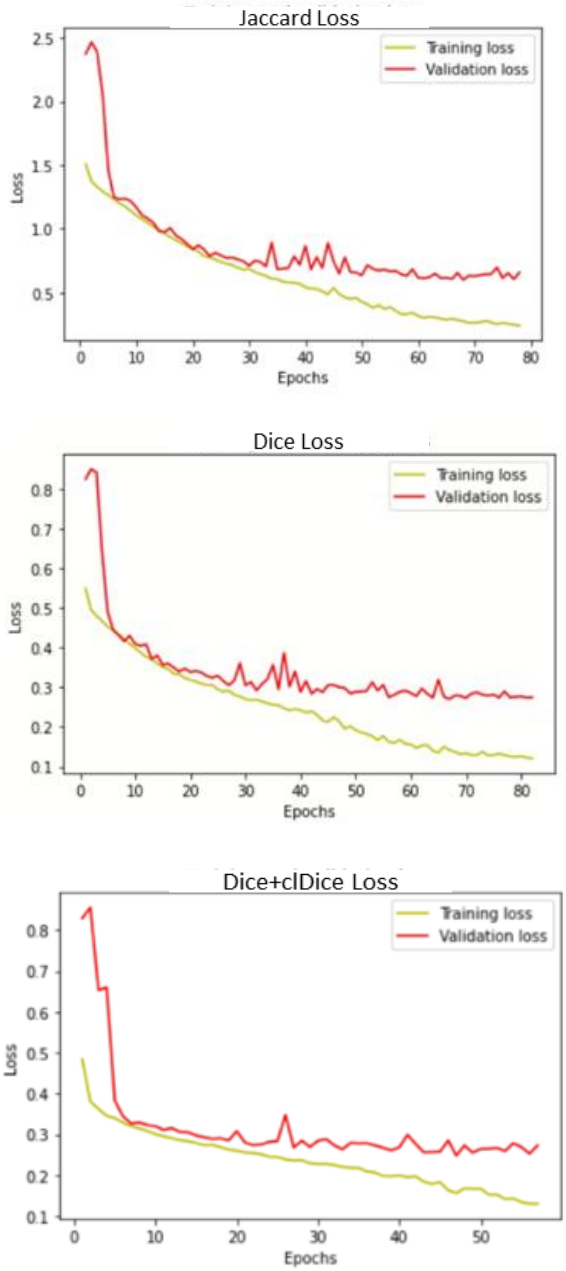


Figure 6.3. *Training and Validation loss curves. From top row to bottom row: Jaccard loss, Dice loss, Dice+clDice loss.*

Loss function	Dice Score	IoU Score	Precision	Recall	Disconnectivity
Jaccard	74.17	61.00	82.07	76.58	3.35
Dice	74.72	61.51	80.19	79.07	3.89
Dice+clDice	74.51	61.17	80.87	77.95	**2.87**

Table 6.1. *Evaluation metrics of 3D U-Net trained using different loss functions, averaged over the 54 volumes in the Test Set.*

All the trainings required approximately the same time, because although the models trained with Jaccard and Dice loss converged later than the model trained with Dice+clDice loss (around 80 against less than 60 epochs), the time required to complete each epoch was higher using the Dice+clDice loss, with 10 minutes required against the 7 minutes required with IoU and Dice loss.

As it can be seen from Table 6.1, the loss used does not significantly affect the volumetric metrics such as Dice or IoU, with a difference of maximum 0.6% between the better performing model and the "worst". On the other hand, it is worth noticing how the model trained using the combination of Dice and clDice, which tries to preserve the vascular topology, yields a less disconnected segmentation, compared to the other two losses used. Therefore, given the fact that our reference segmentations might not be perfectly accurate, we chose as best performing model the one that gives more connected segmentation, trained with Dice+clDice loss, and we retrained the same model, applying data augmentation to the data in the training set. The results obtained are shown in Figure 6.4 and Table 6.2.

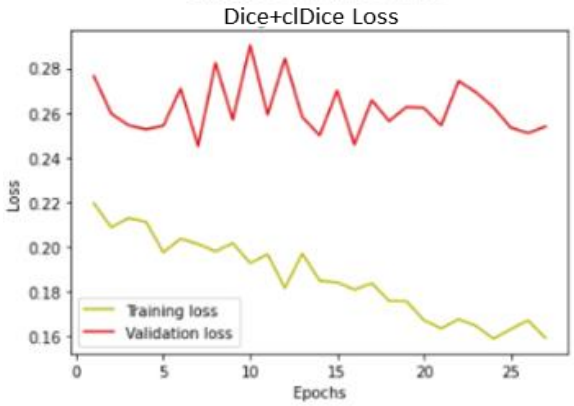


***Figure 6.4**. Training and Validation loss curves for 3D U-Net trained using Dice+clDice loss and retrained using data augmentation on the training set.*

Loss function	Dice Score	IoU Score	Precision	Recall	Disconnectivity
Dice+clDice	74.19	60.50	78.58	79.07	3.05

***Table 6.2**. Evaluation metrics of 3D U-Net trained applying data augmentation on the dataset using the best performing model chosen from above, averaged over the 54 volumes in the test set.*

This way we tried to provide new information to the model that already converged, hence speeding up the convergence time (the model converged after only 7 epochs and continued training for other 20 epochs which was the patience set with EarlyStopping in this case).

The comparison of the results obtained applying data augmentation with the ones obtained without it seems to prove the consideration made before, according to which, in the case of this segmentation task with big size volumes and the repeatability of vessel features, the size of our dataset would be enough for the models to learn.

As discussed in Section 4.1, we also tested the 3D U-Net on our dataset using the original architecture described in [24], to see if the higher number of filters can give better performances, and to do so we trained it on subvolumes of 128x128x96 pixels for memory limitations, randomly split between Training, Validation and Test Set, with the same 50-25-25 ratio used in the other experiments. The results are shown in Figure 6.5 and Table 6.3.

Loss function	Dice Score	IoU Score	Precision	Recall	Disconnectivity
Dice+clDice	73.94	60.19	80.94	76.06	3.67

***Table 6.3**. Evaluation metrics of 3D U-Net trained on the original training set using the architecture described in [24] with a higher number of filters, averaged over the 54 volumes in the test set.*

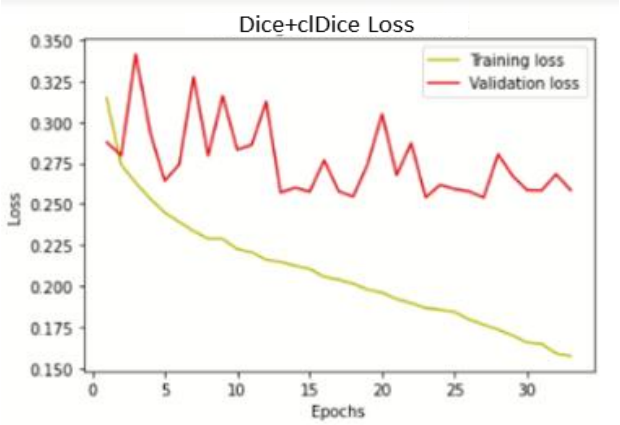


Figure 6.5. *Training and Validation loss curves for 3D U-Net trained using Dice+clDice loss using using the architecture with higher number of filters.*

The evaluation metrics shown are computed on the subvolumes after they have been merged to form again the original volume. It looks like the metrics do not change, while the disconnectivity increase a bit, probably due to the process of merging the subvolumes.

This 3D U-Net architecture is more computationally complex, requiring more time for training compared to the one with less filters used previously. Moreover, this architecture trained on the subvolumes, seems to be more sensitive to artifacts and noise in the image, compared to the one trained on the full volumes. This probably indicates the loss of information caused by the downsampling of already smaller volumes.

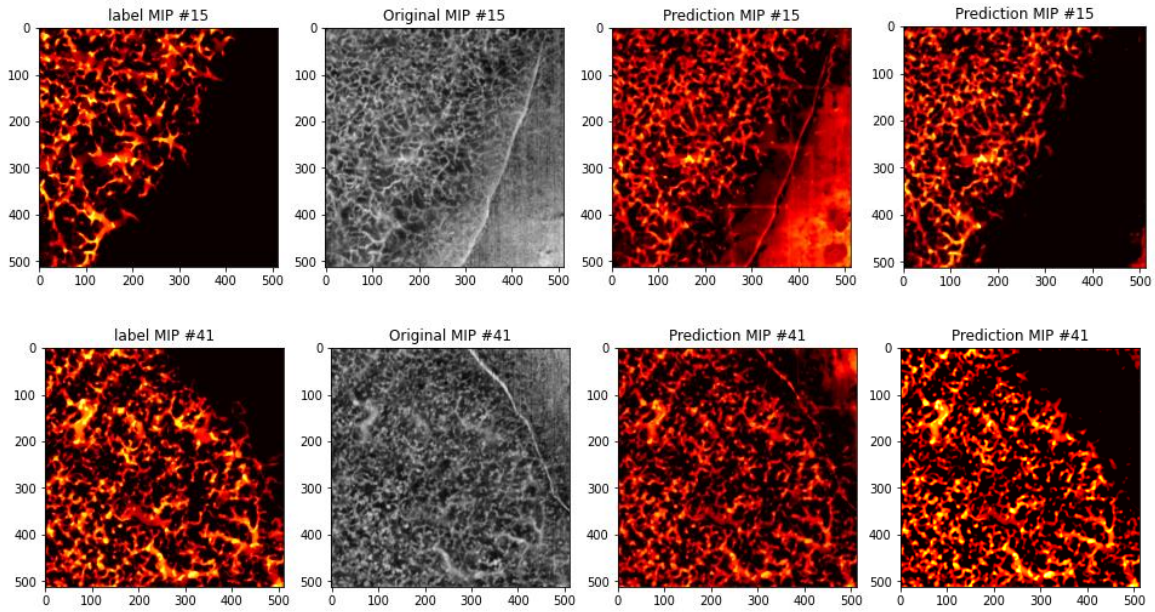


Figure 6.6. *From left to right: ground truth, grayscale volume, prediction of 3D U-Net with more filters trained on subvolumes, prediction of 3D U-Net with less filters trained on whole volumes. The model trained on the whole volumes seems to be less sensitive to noise and artifacts.*

Therefore, the best performing 3D U-Net chosen is the one trained using the Dice+clDice loss on the original training set.

6.2 3D Res-Unet results

Like in the previous section, here we present the results obtained with the 3D Res-Unet trained using different losses and using data augmentation on our OCTA skin volumes.

The learning rate, optimizer and maximum number of epochs are the same as in 3D U-Net, although in this case we use 7-8 epochs as patience for EarlyStopping, since using ResNet34 as a backbone allows the model to converge in few epochs, although increasing the time required for each epoch.

In Figure 6.7 the training and validation loss curves are shown, for each loss used, and in Table 6.4 the evaluation metrics are presented.

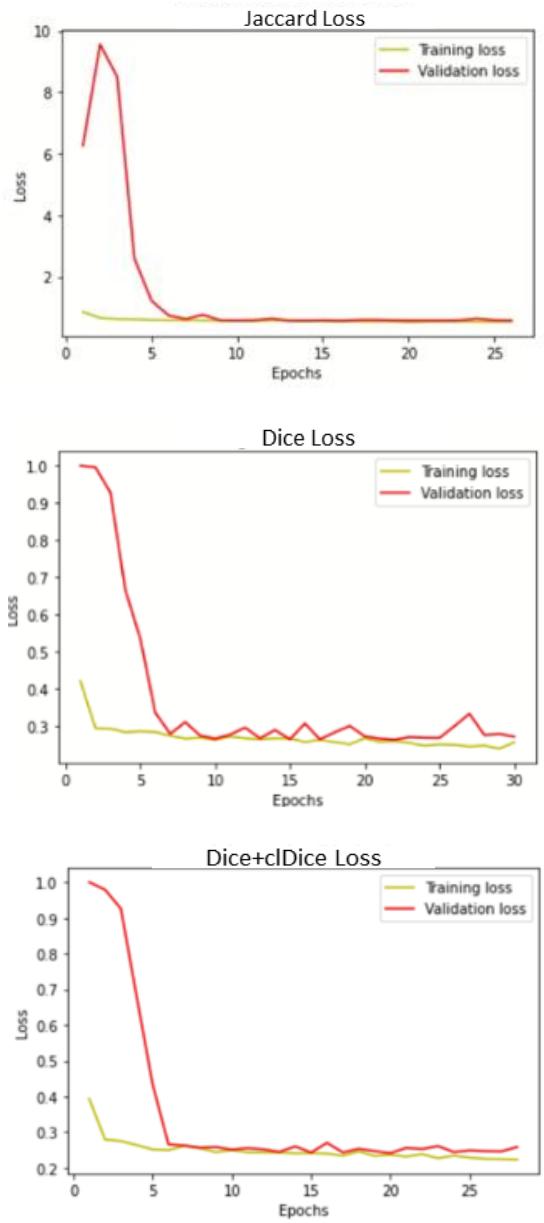


***Figure 6.7**. Training and validation loss curves. From top row to bottom row: Jaccard loss, Dice loss, Dice+clDice loss.*

Loss function	Dice Score	IoU Score	Precision	Recall	Disconnectivity
Jaccard	74.92	61.42	82.30	76.69	3.48
Dice	74.40	60.58	75.26	82.98	3.41
Dice+clDice	74.43	60.60	77.95	79.94	**3.01**

Table 6.4. *Evaluation metrics of 3D Dense-Unet trained using different loss functions, averaged over the 54 volumes in the Test Set.*

The same procedure with data augmentation is applied also here, therefore we trained the best performing model with new artificial data, to check whether the model can improve its performances by training on "new" data. Following the same ideas discussed in the previous section we chose as best model the one yielding less disconnected segmentation, that is the one trained on the combination of Dice and clDice loss. The results are shown below.

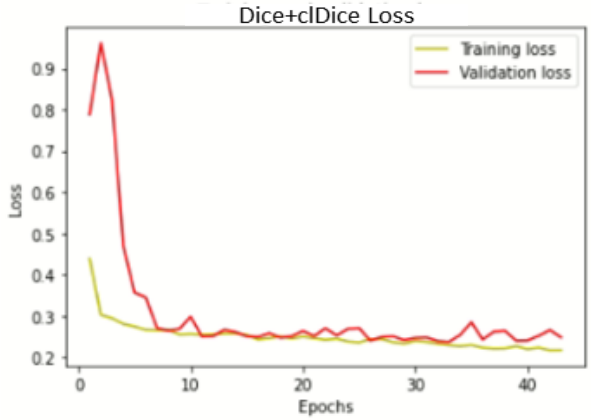


Figure 6.8. *Training and Validation loss curves for 3D Res-Unet trained using Dice+clDice loss using Data Augmentation on the Training Set.*

Loss function	Dice Score	IoU Score	Precision	Recall	Disconnectivity
Dice+clDice	73.77	59.69	74.27	82.74	3.05

Table 6.5. *Evaluation metrics of 3D Res-Unet trained applying Data Augmentation on the dataset using the best performing model chosen from above, averaged over the 54 volumes in the Test Set.*

Here we decided to recreate the architecture of the best 3D Res-Unet configuration, using therefore Dice+clDice loss, and to train it from scratch on the augmented dataset, in order to check different approaches to data augmentation.

However, from the results shown in Table 6.5 it seems like also in this case the data augmentation did not improve the performances.

Therefore, the best performing 3D Res-Unet chosen is the one trained using the Dice+clDice loss on the original training set.

6.3 3D Dense-Unet results

The same procedure followed in the previous experiments was applied also with the 3D Dense-Unet. Below we show the results using the different loss functions.

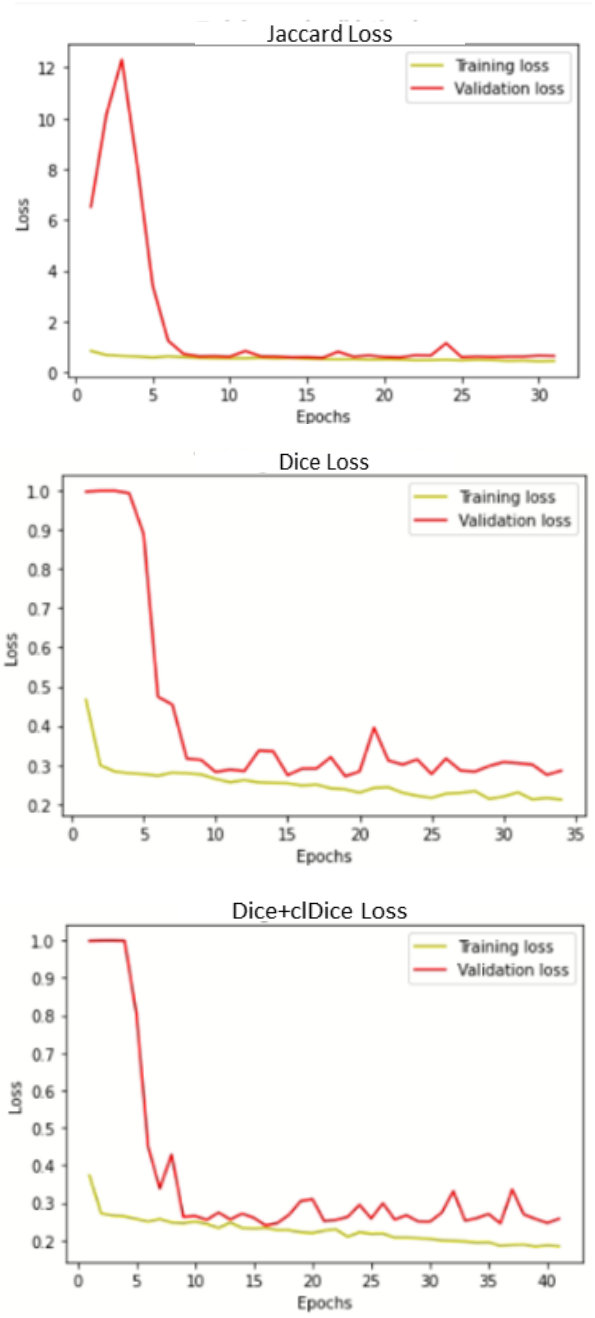


***Figure 6.9**. Training and validation loss curves. From top row to bottom row: Jaccard loss, Dice loss, Dice+ clDice loss.*

Loss function	Dice Score	IoU Score	Precision	Recall	Disconnectivity
Jaccard	74.38	60.56	77.30	80.78	3.10
Dice	73.96	60.02	77.87	79.25	3.64
Dice+clDice	73.91	59.80	76.01	80.83	**2.67**

***Table 6.6**. Evaluation metrics of 3D Dense-Unet trained using different loss functions, averaged over the 54 volumes in the test set.*

As before, we retrained the best performing model, which is the one trained using Dice+clDice loss, applying data augmentation to our Training Set. The results are shown below.

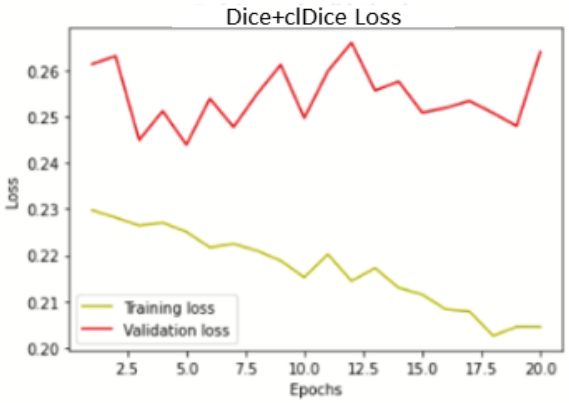


***Figure 6.9**. Training and Validation loss curves for 3D Dense-Unet trained using Dice+clDice loss and retrained using Data Augmentation on the Training Set.*

Loss function	Dice Score	IoU Score	Precision	Recall	Disconnectivity
Dice+clDice	74.07	60.25	79.86	77.46	2.79

***Table 6.7**. Evaluation metrics of 3D Dense-Unet trained applying data augmentation on the dataset using the best performing model chosen from above, averaged over the 54 volumes in the test set.*

In this case we retrained the same 3D Dense-Unet previously trained with Dice+clDice loss, but we reduced the learning rate by a factor of 10, to check if a lower learning rate could help learning a model that has already converged.

Also in this case the data augmentation did not lead to any improvement in the evaluation metrics further strengthening our initial consideration on our dataset size.

Therefore, the best performing 3D Dense-Unet chosen is the one trained using the Dice+clDice loss on the original training set.

6.4 Comparative Analysis

In this section we provide a recap of the best results previously obtained with the three different architectures, and we compare them to better understand the strengths and weaknesses of each one, and how to combine them in order to improve the results, which will be described in Section 6.6.

In the previous sections we presented the results obtained with different architectures, loss functions and techniques such as data augmentation. Analyzing the results, it emerged that in all our experiments, the best performing configuration was the DNN architecture (3D U-Net, 3D Res-Unet, 3D Dense-Unet) trained using the combination of Dice loss and clDice loss, which preserves more the topology of the vessels, and without the need for data augmentation on the training set, likely due to the big size of our volumes and the repetitiveness of the vessels features.

We also showed how the use of a higher number of filters in the 3D U-Net architecture, and the subdivision in smaller sub-volumes did not lead to any significant improvement on the evaluation metrics.

In the Table below the best results for each architecture are shown (using Dice+clDice loss and no data augmentation).

DNN	Dice Score	IoU Score	Precision	Recall	Disconnectivity
3D U-Net	74.51	61.17	80.87	77.95	2.87
3D Res-Unet	74.43	60.60	77.95	79.94	3.01
3D Dense-Unet	73.91	59.80	76.01	80.83	2.67

***Table 6.8**. Evaluation metrics of the best DNNs configuration, using Dice+clDice loss and no Data Augmentation, averaged over the 54 volumes in the test set.*

It looks like overall the results obtained with these three DNNs are similar to each other, although the 3D Dense-Unet seems to be the best one at giving less disconnected segmentations.

It must be noted though, that the 3D Dense-Unet is also the fastest DNN among the three tested in our work. Below we provide the time required for each model.

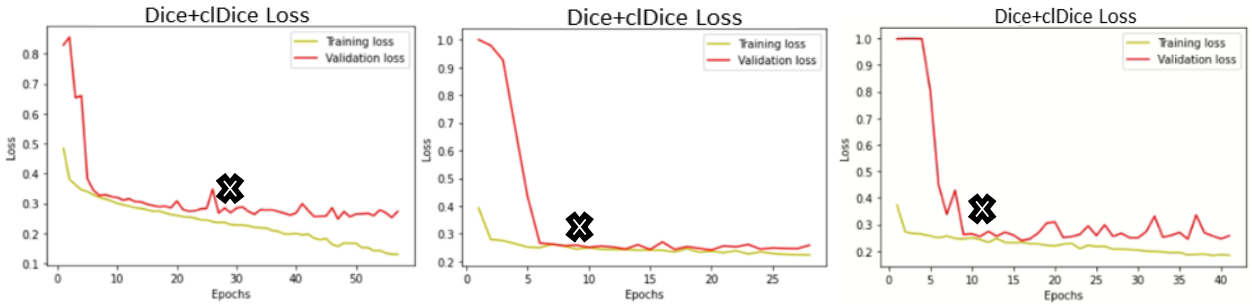


***Figure 6.10**. Training and Validation loss curves for 3D U-Net, 3D Res-Unet and 3D Dense-Unet respectively using Dice+clDice loss and no data augmentation on the training set. Highlighted is the number of epochs where the model approximately converges.*

DNN	Time per epoch (s)	Number of Epochs to converge	Total Training Time
3D U-Net	~ 600	~30	~ 5 hours
3D Res-Unet	~ 1900	~10	~ 5 hours
3D Dense-Unet	~ 600	~10	~ 2 hours

***Table 6.9**. Time required approximately for each DNN to converge.*

Therefore, we could say that the 3D Dense-Unet is the DNN that performs better on our dataset, considering the training time and the disconnectivity metric.

However, by observing the predictions of each model and their metrics values, it is noticeable how the three different models perform differently on different samples indicating the possibility to improve the performances through Ensemble Learning techniques, described in Section 4.3. In particular, it has been observed that out of the 54 samples in the test set, the 3D U-Net yielded segmentations with higher Dice Scores in 20 volumes, and the 3D Res-Unet and the 3D Dense-Unet gave the predictions with the higher scores in 17 volumes each.

6.5 Dual-channel input results

Here the results obtained training the three architectures described until now on the dual-channel volumes are discussed.

The training set is comprised of the same 107 volumes obtained from the split like in the other experiments with the ratio 50-25-25, to which have been added (concatenated) the corresponding Meijering filtered volumes, giving input volumes with shape 512x512x96x2.

The Meijering filter has been applied at different scales to each input volume, with the goal of highlighting both bigger and smaller vessels, and also to reduce the noise in the volumes, since from the previous predictions obtained with the original volumes it was not observed a trend in segmenting better the bigger or smaller vessels, while the noise affected various samples and the predictions on them.

Hence, we trained each of the three DNNs architectures proposed until now, using again Dice+clDice loss and no data augmentation, being the better configuration emerged from the previous experiments, on the dual-channel inputs.

Below are shown the results for each DNN.

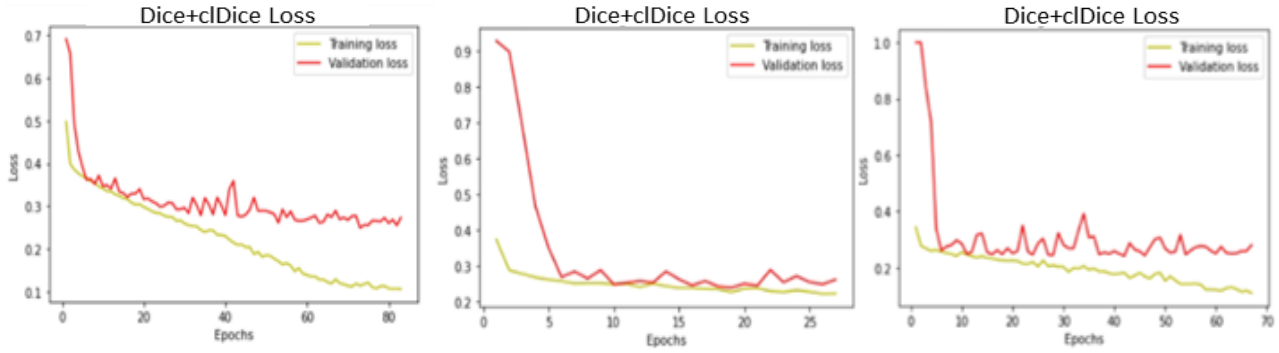


***Figure 6.11**. Training and Validation loss curves for 3D U-Net, 3D Res-Unet and 3D Dense-Unet respectively using Dice+clDice loss and no data augmentation on the dual-channel training set. Highlighted is the number of epochs where the model approximately converges.*

DNN	Dice Score	IoU Score	Precision	Recall	Disconnectivity
3D U-Net	74.78	61.32	80.27	78.56	2.77
3D Res-Unet	74.00	60.09	76.88	80.08	2.94
3D Dense-Unet	73.36	59.64	80.10	75.95	2.45

***Table 6.10**. Evaluation metrics of the best DNNs configuration, on the dual-channel training set using Dice+clDice loss and no data augmentation, averaged over the 54 volumes in the test set.*

Using the dual-channel inputs did not yield any significant improvement overall on the evaluation metrics, except for the disconnectivity, which is the only metric that improved for all the three DNNs compared to their best results on the original training set.

Regarding the training time required for each model the same discussion of earlier applies, with 3D Dense-Unet being the fastest model to converge.

Also in this case we noticed how the three models perform differently among the samples in the test set, indicating the possibility to improve the performances with Ensemble Learning techniques.

6.6 Ensemble Learning results

After analyzing all the previous results, it appears that the three model architectures trained using the combination of Dice and clDice loss on the Training Set, without data augmentation and using the dual-channel inputs, are the configurations that give the best results, considering both the evaluation metrics and the training time required, which slightly increases when using data augmentation, while it does not increase with the dual-channel inputs, although applying the Meijering filter at different scales to all the volumes requires additional computing time.

Therefore, as a further step to try to improve the segmentation performances, we applied the two Ensemble Learning techniques described in Section 4.3, Majority Voting and Dynamic Model Selection, to these three model architectures that yielded the best results, mentioned in Table 6.10.

In the Table below the evaluation metrics obtained with these two techniques are provided.

	Dice Score	**IoU Score**	**Precision**	**Recall**	**Disconnectivity**
Majority Voting	74.85	61.35	79.94	78.77	2.56
Dynamic Model Selection	76.77	63.65	80.82	80.09	2.76

***Table 6.11**. Evaluation metrics obtained using two Ensemble Learning techniques on the three model architectures trained using Dice+clDice loss on the dual-channel training set, averaged over the 54 volumes in the test set.*

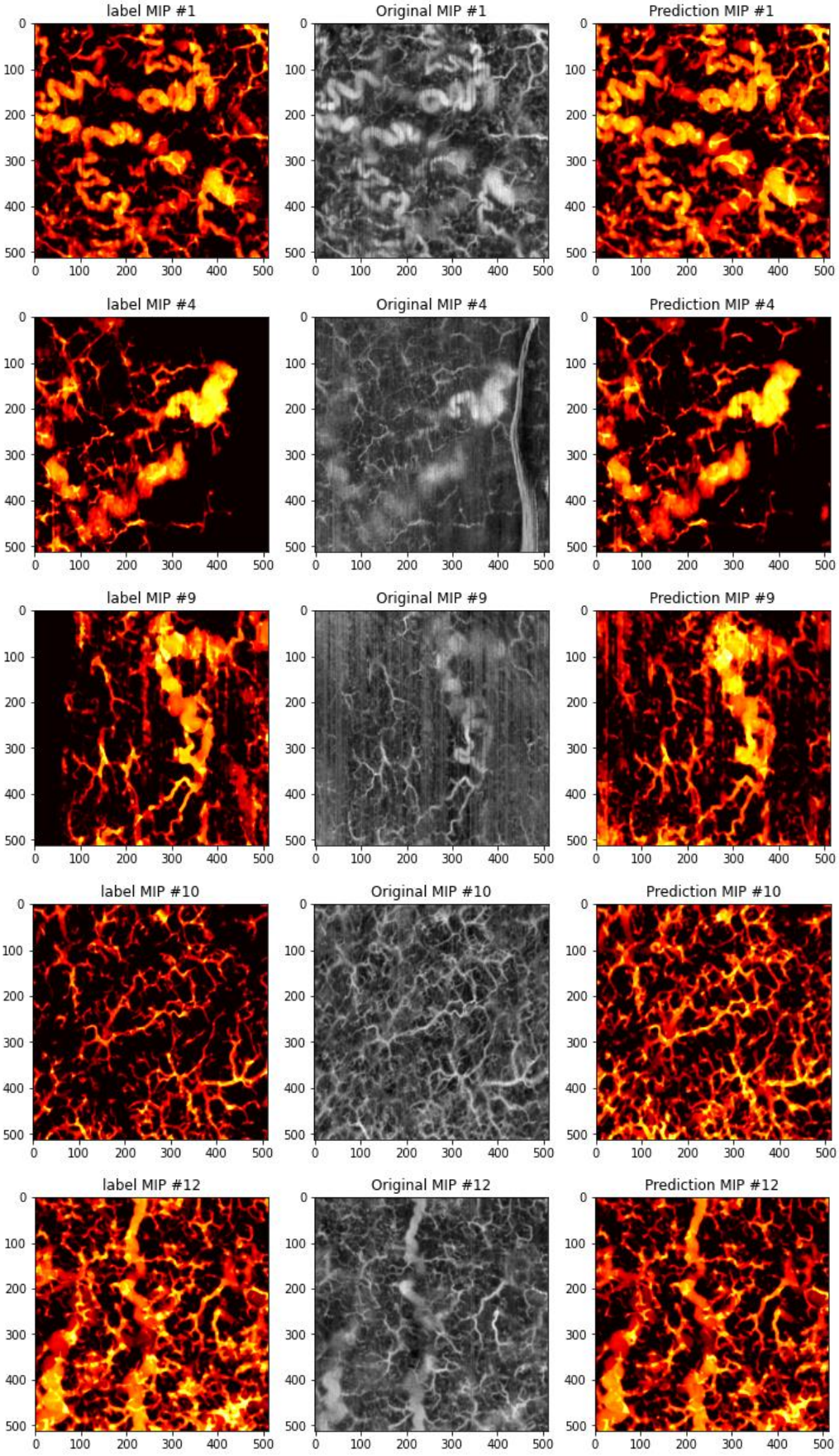
The results on Dice Score and IoU Score obtained using Dynamic Model Selection improved of more than 2% over the best metrics obtained with the three model architectures individually, which was expected given that the criteria used to select the prediction of each sample was the one with the highest Dice Score. The disconnectivity of 2.76 worsened compared to the 2.45 obtained with the 3D Dense-Unet with dual-channel inputs.

On the other hand, the Majority Voting technique, which slightly improves the results compared to the individual model architectures, does not improve significantly like Dynamic Model Selection, although the disconnectivity is lower, with 2.56.

At the same time, it should be considered that the metrics computed with Majority Voting take into account the predictions of at least two models for each pixel, giving therefore more reliable predictions.

However, we decided to choose as best performing technique the Dynamic Model Selection in our task, given the 2% difference in Dice Score and IoU score.

In Figure 6.12 some examples of prediction from the Dynamic Model Selection are shown.


label MIP #1
Original MIP #1
Prediction MIP #1
label MIP #4
Original MIP #4
Prediction MIP #4
label MIP #9
Original MIP #9
Prediction MIP #9
label MIP #10
Original MIP #10
Prediction MIP #10
label MIP #12
Original MIP #12
Prediction MIP #12

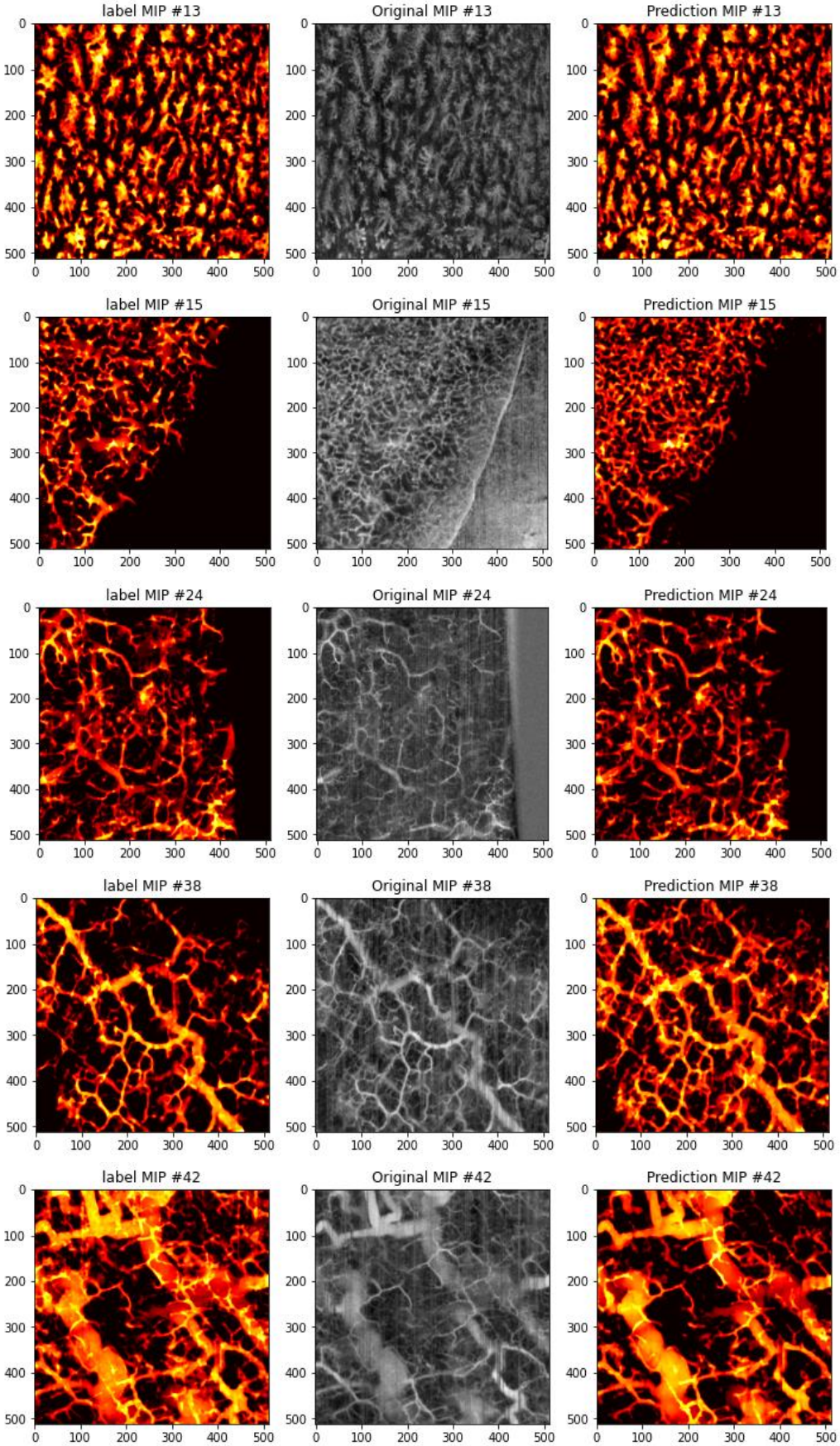

label MIP #13
Original MIP #13
Prediction MIP #13
label MIP #15
Original MIP #15
Prediction MIP #15
label MIP #24
Original MIP #24
Prediction MIP #24
label MIP #38
Original MIP #38
Prediction MIP #38
label MIP #42
Original MIP #42
Prediction MIP #42

***Figure 6.12**.En face view of MIP of different predictions on test set samples with the Dynamic Model Selection. Figures #1,4,12,13,24,42 provide accurate segmentations, both visually and metric-wise (Dice>80%), despite the presence of different artifacts in the volume (#4,24) and the variability in the vascular structure due to different types of skin, healthy or CVI (#1,12,13,42). On the other hand, among the volumes that give low quality segmentation there are figures #9,10,15,38, although from a visual inspection it can be seen as the low metric values are most likely due to the inaccuracy of the manual segmentation (#10,15,38) or to a low-quality volume such as #9, affected by artifacts and noise.*

Among these examples presented it has been observed that the low value of evaluation metrics in some predicted segmentations are due to the inaccuracy of the reference segmentations in the majority of case, rather than in a poor segmentation by the models.

The results obtained using the Dynamic Model Selection approach demonstrate that our models are capable of accurately segmenting skin vessels in a variety of different scenarios, with different size vessels and both healthy and diseased skin. Several of the maximum intensity projection (MIP) en face views presented in our study provide accurate segmentations, with Dice scores exceeding 80%, and in the case of low metric values, it is not always indicative of low quality segmentation, but more of a inaccurate ground truth.

Finally, both a visual inspection of the predicted segmentations and the disconnectivity values computed for some of the segmented volumes, suggest that post-processing might be necessary to improve the overall segmentation performance, in particular to obtain less disconnected segmentations, which will be discussed in the next paragraph.

6.7 Post-processing

As last effort to obtain the best possible segmentations, trying to balance both volumetric metrics such as Dice Score and IoU Score, and the continuity of the vascular network we applied two different post-processing techniques to the predictions obtained with the Dynamic Model Selection on the dual-channel inputs.

The two techniques applied subsequently are the removal of small objects and the binary closing.

First, we removed small segmented objects based on their size, removing the ones with a size smaller than 200 pixels, where pixels are considered belonging to the same object depending on the connectivity that defines the neighborhood of a pixel.

Despite the difficulty to visualize vascular networks of this size, in Figure 6.13 we provide two examples of both the ground truth and the predicted segmentation after the removal of small objects, in which separated vessels are labelled with different colors.

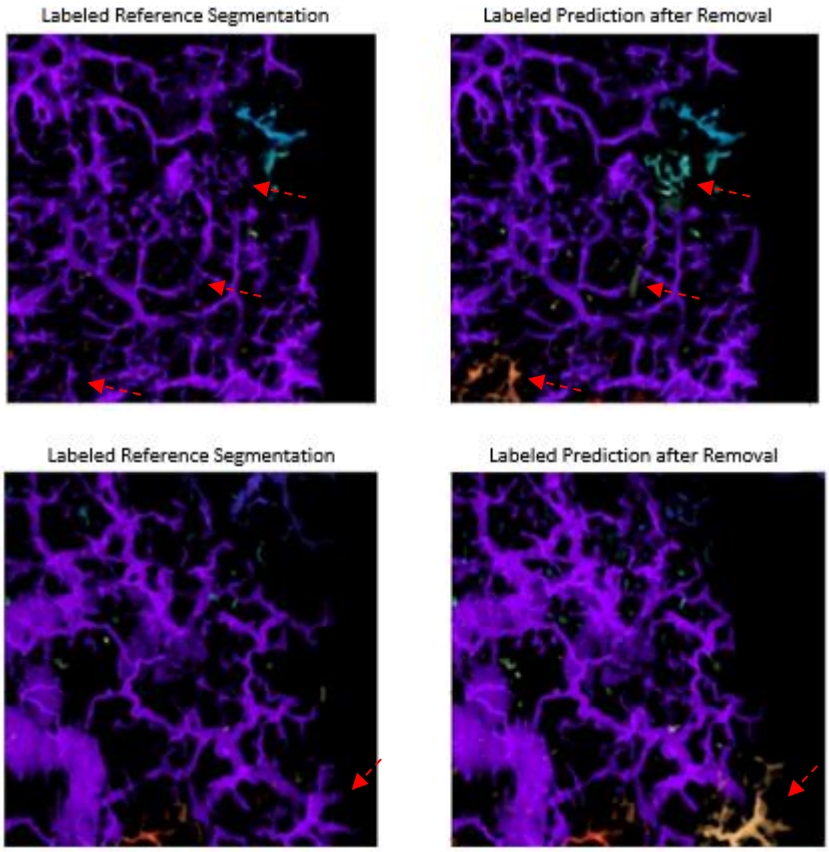


***Figure 6.13**. Comparison of ground truth and predicted segmentation after the removal of small objects from for two predicted samples in the Test Set. The red arrows point to vessels that are connected to the main network in the ground truth but disconnected in the predictions.*

Although this technique seems to be highly effective in the cleaning of the segmentation, proven by the average disconnectivity value in the test set after the removal of small objects (Table 6.14) which decreases compared to the value before removal, there are still some big branches like the ones pointed in Figure 6.13 which are not connected to the main vascular network, like it is instead in the ground truth.

	Dice Score	**IoU Score**	**Precision**	**Recall**	**Disconnectivity**
Before removal	76.77	63.65	80.82	80.09	2.76
After removal	76.74	63.63	80.92	79.97	**1.11**

***Table 6.12**. Evaluation metrics on the test set with the prediction obtained from Dynamic Model Selection on dual-channel inputs, before and after the removal of small objects.*

Therefore, we proceeded to apply a post-processing technique called binary closing, with the intent to try to reconnect at least the bigger branches, without connecting vessels that should not be connected.

Binary closing is a morphological operator that is often used in image processing to remove small holes or gaps in binary images and is defined as a dilation followed by an erosion. Dilation "grows" the image, while erosion "shrinks" it, depending on a structuring element that will be overlapped to each pixel in the original image, to determine whether to activate/turn off each pixel. Given our task we selected a structuring element with the shape of a sphere with radius of 3 pixels, which is approximately the diameter of the small vessels.

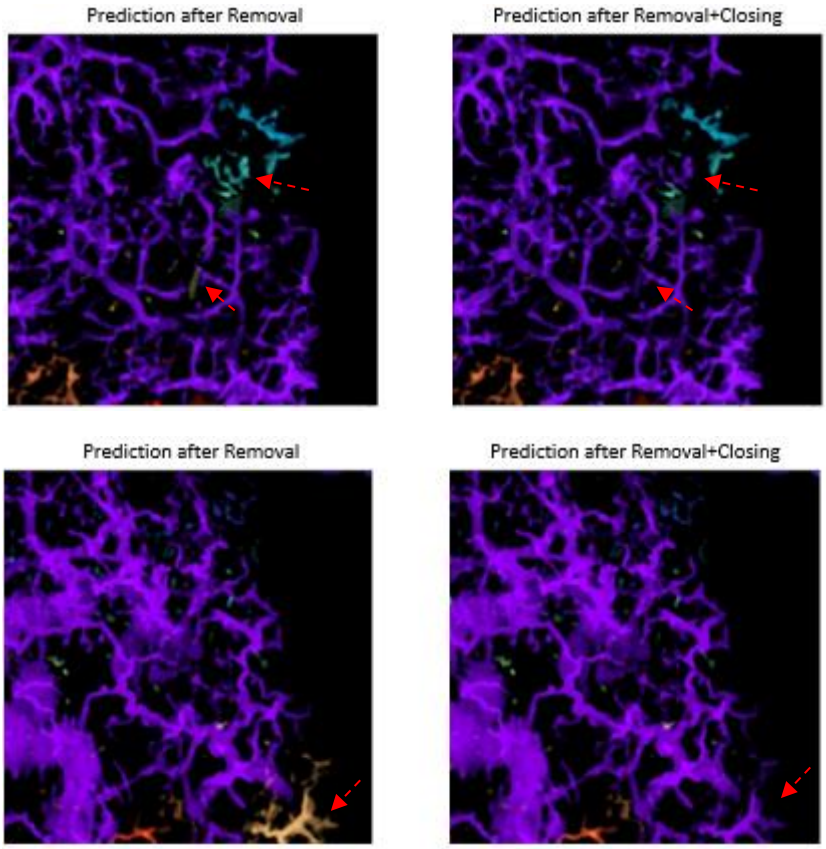


***Figure 6.14**. Comparison of connected vessels in predicted segmentation after the removal of small objects and after closing for the previously shown two samples in the Test Set. Red arrows point to vessels that have been connected after the morphological operation of closing has been applied.*

The overall metrics on the Test Set does not change significantly, with the disconnectivity now being lower than 1, which would mean that the predicted segmentations are less disconnected than the ground truth, but to determine whether this is a good effect or a negative one caused by the wrong connection of vessels, a visual inspection on every volume should be done, which is very laborious and prone to errors.

	Dice Score	IoU Score	Precision	Recall	Disconnectivity
After removal	76.74	63.63	80.92	79.97	1.11
After closing	76.48	63.25	79.75	80.64	**0.80**

***Table 6.13**. Evaluation metrics on the test set with the prediction obtained from Dynamic Model Selection on dual-channel inputs, after the removal of small objects, and after closing.*

From Figure 6.14 it can be seen that the binary closing was successful in unifying some of the big branches that were previously disconnected in the prediction after removal of objects, while they were connected in the ground truth (the ones pointed by the red arrows).

The disconnectivity value lower than 1 means that the predicted segmentations are less disconnected than the ground truth, and this can be seen happening in some samples, shown in the figure below.

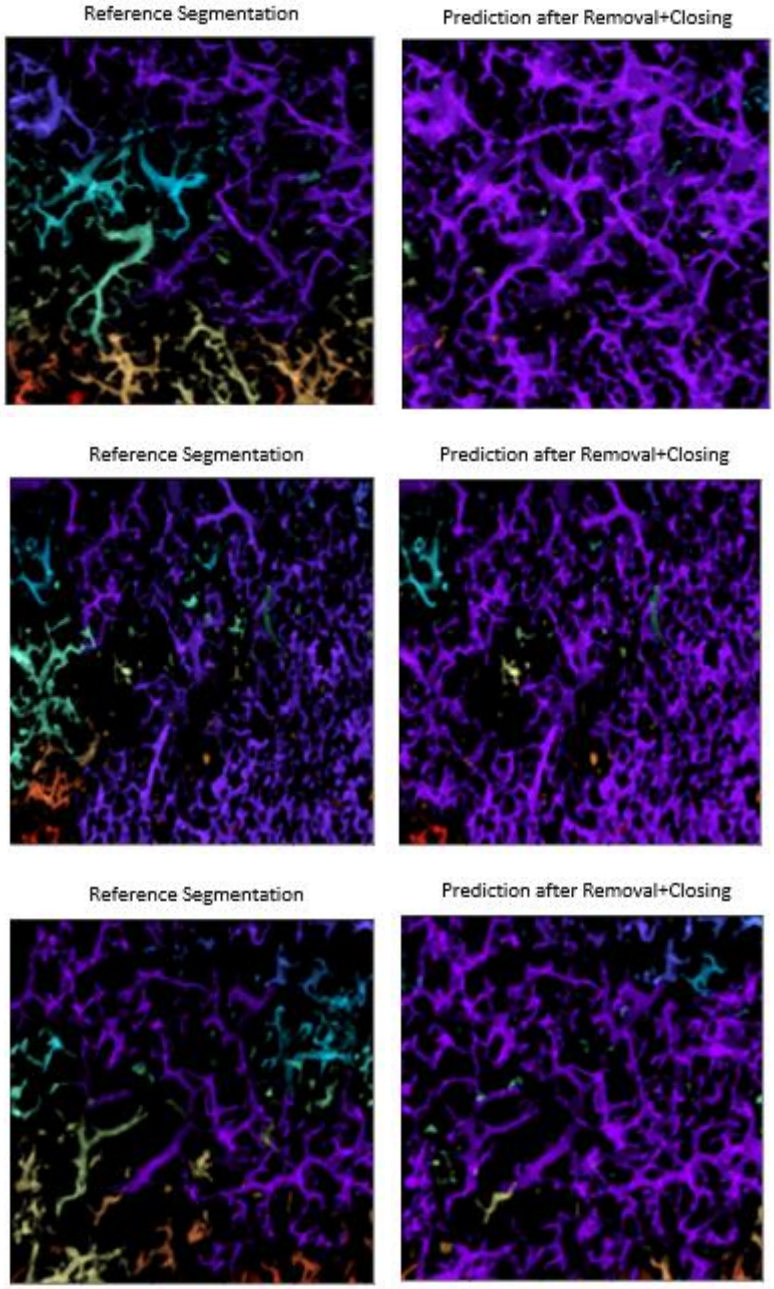


***Figure 6.15**. Examples of predicted segmentations with less disconnections than the ground truths.*

Probably a structuring element with a bigger radius than the one used (3 pixels) could connect also other disconnected vessels but would probably deteriorate the segmentation of volumes with very intricate and small vessels, therefore, we did not use bigger structuring element sizes.

7. Conclusions

Given the possible clinical applications of OCTA both in ophthalmology and dermatology, the advantages of visualizing the 3D segmented vascular network and the challenges of manual segmentation of small vessels in large volumes, we developed an automatic segmentation algorithm using deep learning methods with the goal to provide accurate segmentations of OCTA skin volumes with a FOV of 10x10 mm.

In this study, we have investigated the performance of three different deep learning architectures, 3D U-Net, 3D Dense-Unet and 3D Res-Unet, for vascular segmentation in 3D OCTA skin volumes. Our results have shown that all three architectures performed slightly better when trained with the combination of Dice loss and clDice loss which better preserves the topology of the vessels, and using dual-channel inputs, which included both the original volume and its Meijering filtered version. Although yielding approximately similar evaluation metrics values, the dual-channel inputs give less disconnected segmentations.

When it comes to the evaluation metrics, our findings have shown that the three architectures achieved similar results, although the 3D Dense-Unet presented the lowest disconnectivity value of 2.67, which measures the degree of connectivity between segmented vessel voxels. However, they performed differently on different samples, indicating the possibility to combine their strengths through an Ensemble Learning approach. Indeed, the best results were obtained when we combined the outputs of the three models through Dynamic Model Selection, selecting for each sample the prediction with the highest Dice Score among the predicitions of each model, achieving higher scores on the evaluation metrics.

Finally, we applied two post-processing techniques, removal of small objects and binary closing, which respectively clean the segmentation removing small, isolated regions and connect wrongly disconnected vessels. This led to an improving in the disconnectivity metric, which was also verified through a visual inspection of the segmentations, without affecting the other evaluation metrics.

It is important to notice that the reference segmentation should not be taken as absolute truth in some cases, given the unreliability of the semi-automatic segmentation obtained using Amira and considering the intra-variability of different operators that segmented the OCTA volumes. In fact, we have found that some segmentations with low metrics values were actually due to an inaccurately segmented ground truth, while the models were better at segmenting in some cases. Therefore, it is important to approach the ground truth with a critical eye and to validate it with other methods whenever possible.

Additionally, the volumes that were segmented in this study presented diverse challenges. These included a large FOV (10x10 mm), the presence of noise and artifacts within the volumes, and variations in skin type, including healthy and diseased skin with different CVI stages. These challenges may have affected the performance of the models, which however still looks promising, and further research is needed to investigate their impact on the accuracy and robustness of the segmentation results.

Finally, as future works, it could be possible to compute vascular parameters on the segmentations obtained with the deep learning models and the ground truth to provide valuable information for clinical diagnosis and treatment planning. It may be also useful to test the algorithm on a larger and diverse set of datasets and volumes acquired from different OCT systems. This will help to validate the algorithm's robustness and generalizability across different clinical settings. Furthermore, reference segmentations should be improved to enhance the accuracy of the algorithm. These efforts will further facilitate the clinical implementation of automated segmentation methods for 3D OCTA volumes.

Bibliography

[1] Eleni Nikolopoulou, Massimo Lorusso, Luisa Micelli Ferrari, Maria Vittoria Cicinelli, Francesco Bandello, Giuseppe Querques, Tommaso Micelli Ferrari, "Optical Coherence Tomography Angiography versus Dye Angiography in Age-Related Macular Degeneration: Sensitivity and Specificity Analysis", BioMed Research International, vol. 2018, Article ID 6724818, 7 pages, 2018. https://doi.org/10.1155/2018/6724818

[2] Matsunaga, Douglas R et al. "Optical Coherence Tomography Angiography of Diabetic Retinopathy in Human Subjects." Ophthalmic surgery, lasers & imaging retina vol. 46,8 (2015): 796-805. doi:10.3928/23258160-20150909-03

[3] Tsai, Grace et al. "Optical Coherence Tomography Angiography in Eyes with Retinal Vein Occlusion." Journal of ophthalmic & vision research vol. 13,3 (2018): 315-332. doi:10.4103/jovr.jovr_264_17

[4] Meiburger, Kristen M., et al. "Automatic Segmentation and Classification Methods Using Optical Coherence Tomography Angiography (OCTA): A Review and Handbook." Applied Sciences, vol. 11, no. 20, Oct. 2021, p. 9734. Crossref, https://doi.org/10.3390/app11209734.

[5] Li, Mingchao, et al. "Ipn-v2 and octa-500: Methodology and dataset for retinal image segmentation." arXiv preprint arXiv:2012.07261 (2020).

[6] Guo, Menglin et al. "Automatic quantification of superficial foveal avascular zone in optical coherence tomography angiography implemented with deep learning." Visual computing for industry, biomedicine, and art vol. 2,1 21. 9 Dec. 2019, doi:10.1186/s42492-019-0031-8

[7] Liu, Mengyang , and Wolfgang Drexler . "Optical coherence tomography angiography and photoacoustic imaging in dermatology." Photochemical & photobiological sciences: Official journal of the European Photochemistry Association and the European Society for Photobiology vol. 18,5 (2019): 945-962. doi:10.1039/c8pp00471d

[8] Deegan, Anthony J et al. "Optical coherence tomography angiography of normal skin and inflammatory dermatologic conditions." Lasers in surgery and medicine vol. 50,3 (2018): 183-193. doi:10.1002/lsm.22788

[9] Meiburger, K. M., Chen, Z., Sinz, C., Hoover, E., Minneman, M., Ensher, J., Kittler, H., Leitgeb, R. A., Drexler, W., & Liu, M. (2019). Automatic skin lesion area determination of basal cell carcinoma using optical coherence tomography angiography and a skeletonization approach: Preliminary results. Journal of biophotonics, 12(9), e201900131. https://doi.org/10.1002/jbio.201900131

[10] Huang D, Swanson EA, Lin CP, Schuman JS, Stinson WG, Chang W, Hee MR, Flotte T, Gregory K, Puliafito CA, et al. Optical coherence tomography. Science. 1991 Nov 22;254(5035):1178-81. doi: 10.1126/science.1957169. PMID: 1957169; PMCID: PMC4638169.

[11] Drexler, W., Liu, M., Kumar, A., Kamali, T., Unterhuber, A., & Leitgeb, R. A. (2014). Optical coherence tomography today: speed, contrast, and multimodality. Journal of biomedical optics, 19(7), 071412. https://doi.org/10.1117/1.JBO.19.7.071412

[12] Bezerra HG, Costa MA, Guagliumi G, Rollins AM, Simon DI. Intracoronary optical coherence tomography: a comprehensive review clinical and research applications. JACC Cardiovasc Interv. 2009 Nov;2(11):1035-46. doi: 10.1016/j.jcin.2009.06.019. PMID: 19926041; PMCID: PMC4113036.

[13] Maier, T., Sattler, E., Braun-Falco, M., Ruzicka, T., & Berking, C. (2012). High-definition optical coherence tomography for the in vivo detection of demodex mites. Dermatology (Basel, Switzerland), 225(3), 271–276. https://doi.org/10.1159/000345364

[14] Fujimoto, J G et al. “Optical coherence tomography: an emerging technology for biomedical imaging and optical biopsy.” Neoplasia (New York, N.Y.) vol. 2,1-2 (2000): 9-25. doi:10.1038/sj.neo.7900071

[15] Zhe Chen, Mengyang Liu, Michael Minneman, Laurin Ginner, Erich Hoover, Harald Sattmann, Marco Bonesi, Wolfgang Drexler, and Rainer A. Leitgeb, "Phase-stable swept source OCT angiography in human skin using an akinetic source," Biomed. Opt. Express 7, 3032-3048 (2016)

[16] de Carlo, T.E., Romano, A., Waheed, N.K. et al. A review of optical coherence tomography angiography (OCTA). Int J Retin Vitr 1, 5 (2015). https://doi.org/10.1186/s40942-015-0005-8

[17] Spaide, Richard F et al. “Optical coherence tomography angiography.” Progress in retinal and eye research vol. 64 (2018): 1-55. doi:10.1016/j.preteyeres.2017.11.003

[18] Borrelli, Enrico et al. “Quantification of diabetic macular ischemia using novel three-dimensional optical coherence tomography angiography metrics.” Journal of biophotonics vol. 13,10 (2020): e202000152. doi:10.1002/jbio.202000152

[19] Ranjbarzadeh, R., Bagherian Kasgari, A., Jafarzadeh Ghoushchi, S. et al. Brain tumor segmentation based on deep learning and an attention mechanism using MRI multi-modalities brain images. Sci Rep 11, 10930 (2021). https://doi.org/10.1038/s41598-021-90428-8

[20] Zhou, Xiangrong. “Automatic Segmentation of Multiple Organs on 3D CT Images by Using Deep Learning Approaches.” Advances in experimental medicine and biology vol. 1213 (2020): 135-147. doi:10.1007/978-3-030-33128-3_9

[21] Stalling, Detlev et al. “amira: A Highly Interactive System for Visual Data Analysis.” The Visualization Handbook (2005).

[22] Choi, Woo June et al. “Mean-Subtraction Method for De-shadowing of Tail Artifacts in Cerebral OCTA Images: A Proof of Concept.” Materials (Basel, Switzerland) vol. 13,9 2024. 26 Apr. 2020, doi:10.3390/ma13092024

[23] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *International Conference on Medical image computing and computer-assisted intervention*. Springer, Cham, 2015.

[24] Çiçek, Özgün, et al. "3D U-Net: learning dense volumetric segmentation from sparse annotation." International conference on medical image computing and computer-assisted intervention. Springer, Cham, 2016.

[25] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[26] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

[27] Szegedy, Christian, et al. "Going deeper with convolutions." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

[28] Solovyev, Roman, Alexandr A. Kalinin, and Tatiana Gabruseva. "3D convolutional neural networks for stalled brain capillary detection." *Computers in Biology and Medicine* 141 (2022): 105089.

[29] https://github.com/ZFTurbo/segmentation_models_3D

[30] Huang, Gao, et al. "Densely connected convolutional networks." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.

[31] Meijering, E et al. “Design and validation of a tool for neurite tracing and analysis in fluorescence microscopy images.” Cytometry. Part A: the journal of the International Society for Analytical Cytology vol. 58,2 (2004): 167-76. doi:10.1002/cyto.a.20022

[32] Survarachakan, Shanmugapriya, et al. “Effects of Enhancement on Deep Learning Based Hepatic Vessel Segmentation.” Electronics, vol. 10, no. 10, May 2021, p. 1165. Crossref, https://doi.org/10.3390/electronics10101165.

[33] Yeung, Michael, et al. "Unified Focal loss: Generalising Dice and cross entropy-based losses to handle class imbalanced medical image segmentation." Computerized Medical Imaging and Graphics 95 (2022): 102026.

[34] Tiulpin, Aleksei, et al. "Deep-learning for tidemark segmentation in human osteochondral tissues imaged with micro-computed tomography." Advanced Concepts for Intelligent Vision Systems: 20th International Conference, ACIVS 2020, Auckland, New Zealand, February 10–14, 2020, Proceedings. Cham: Springer International Publishing, 2020.

[35] Iglovikov, Vladimir, and Alexey Shvets. "Ternausnet: U-net with vgg11 encoder pre-trained on imagenet for image segmentation." arXiv preprint arXiv:1801.05746 (2018).

[36] Shit, Suprosanna, et al. "clDice-a novel topology-preserving loss function for tubular structure segmentation." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021.