FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Characterization of Retinal Fluid in OCT Images

David Castanho Terroso

WORKING VERSION

Mestrado em Engenharia Biomédica

Supervisor: Prof. Tânia Melo

Co-Supervisor: Prof. Ana Maria Mendonça

June 24, 2025

Resumo

Abstract

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework to achieve a better and more sustainable future for all. It includes 17 goals to address the world's most pressing challenges, including poverty, inequality, climate change, environmental degradation, peace, and justice.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis.

The specific Sustainable Development Goals mentioned have the following names:

SDG 3 Ensure healthy lives and promote well-being for all at all ages

SGD	Target	Contribution	Performance Indicators and Metrics
3	3.b		
	3.c		
	3.d		

Acknowledgements

David Castanho Terroso

"Our greatest glory is not in never falling, but in rising every time we fall"

Confucius

Contents

1 Introduction **1**

2 Literature Review **5**
2.1 Fluid Segmentation . . . 5
2.1.1 Search Strategy . . . 5
2.1.2 Literature Review . . . 6
2.2 Intermediate Slice Synthesis . . . 9
2.2.1 Architectures . . . 9

3 Methods **18**
3.1 Dataset . . . 18
3.2 Experiments . . . 19
3.2.1 Cross-validation . . . 20
3.2.2 Fluid Segmentation . . . 21
3.2.3 Intermediate Slice Synthesis . . . 28
3.2.4 Fluid Volume Estimation . . . 35

4 Results and Discussion **37**
4.1 Fluid Segmentation . . . 37
4.1.1 Experiment 1 . . . 38
4.1.2 Experiment 2 . . . 53
4.2 Intermediate Slice Synthesis . . . 61
4.2.1 Experiment 3 . . . 61
4.2.2 Experiment 4 . . . 61
4.3 Fluid Volume Estimation . . . 61
4.3.1 Experiment 5 . . . 61
4.3.2 Experiment 6 . . . 61

References **62**

A Variability of the Segmentation U-Net **69**

List of Figures

1.1 OCT B-scans (A), when taken at a fixed distance along the azimuthal axis, form a volume of the posterior segment of the eye (B) [12]. . . . 2
1.2 The three distinct fluid types on an OCT B-scan: IRF in red, SRF in green, and PED in blue [2]. . . . 3
1.3 OCT scan of the retinal layers [13]. . . . 3

2.1 Grouping of the articles included in the literature review. . . . 6
2.2 Example of a CNN architecture used in fluid binary segmentation. Image A depicts the neural network architecture. B shows the used multi-scale block, while C and D exhibit the residual convolutional blocks [19]. . . . 7
2.3 Example of a framework that includes delimitation of the retinal layer and a relative distance map (left side). The generated map is included in the segmentation network (denominated ICAF-Net, by the authors) [32]. . . . 9
2.4 López-Varela et al. [14] training process. . . . 10
2.5 Framework developed by Xia et al. [41]. . . . 12
2.6 Architecture of the method developed by Zhang et al. [44]. . . . 13
2.7 Pipeline of the methodology utilized by Fang et al. [46]. . . . 14
2.8 Pipeline that describes the RIFE framework. The student model attempts to generate the intermediate frame, while the teacher refines the frame generated by the student so that it looks more similar to the middle frame. The results from both networks are evaluated on the reconstruction loss [57]. . . . 16
2.9 Pipeline representing the framework developed by Tran and Yang [58]. The generator that generates the intermediate frame is represented by G, while its discriminator is labeled D. The pix2pix generator is denoted by G_RN and the discriminator is represented by D_RN. x_{n-1} and x_{n+1} respectively represent the previous and following frames of the one that is being generated, x_n. y_n is the image generated by the first generator, while y'_n is the image refined by the pix2pix network [58]. . 17

3.1 U-Net architecture [36]. . . . 22
3.2 Cirrus B-scan (left), fluid masks overlay (middle) with IRF in red, SRF in green, and PED in blue, and the ROI mask overlaid in purple (right). The red bounding box signals a possible 256 × 128 patch that could be extracted. . . . 24
3.3 Cirrus B-scan and its respective three patches of shape 496 × 512. . . . 25
3.4 B-scan of the retinal layers in different patients, using Cirrus (left) and Spectralis (right) devices. In Cirrus, the retinal layers appear much larger than in Spectralis. 25
3.5 Four vertical patches of shape 496 × 128 extracted from a Cirrus B-scan. . . . 26
3.6 Seven vertical patches of shape 496 × 128 extracted from a Cirrus B-scan. . . . 26
3.7 Thirteen vertical patches of shape 496 × 128 extracted from a Cirrus B-scan. . . 27

3.8 Scheme explaining the input data of the generative models. Each frame refers to B-scan from an OCT volume. Extracted from Tran and Yang [58]. 29
3.9 Example of a GAN framework, where $\mathscr{D}$ is the discriminator and $\mathscr{G}$ is the generator [71]. 30
3.10 Patches with shape 64 $\times$ 64 extracted from a Cirrus B-scan which was resized to 496 $\times$ 512. 31
3.11 Architecture of the generator used in the GAN. It has a contracting and an expanding path, making it a U-Net like network [58]. 32

4.1 Example of a patch extracted from a Cirrus OCT volume used in Experiment 1.1. In this patch, while the background is noticeable due to its darker shade, the choroid is harder to be identified by an observer (or a model) due to the lack of context. 39
4.2 Example of a poor segmentation made by the model trained in Run 1 (right). In the left, the GT mask for the same image is shown. 40
4.3 Example of the segmentation done by the model trained in Run 9 (right) and its respective GT mask (left). The B-scan segmented is the same as in Figure 4.2. . . 41
4.4 Example of the segmentation done by the model trained in Run 12 (right) and its respective GT mask (left). It is noticeable that the model confuses the choroid with the retina, as segmentation of IRF and PED is performed in the choroid. . . 41
4.5 Training and validation losses in Run 13 (left) and Run 17 (right). Despite reaching the validation loss minimum in a similar number of epochs and having comparable training and validation loss curves, the performance is really different in Table 4.3. 43
4.6 Predicted masks by the models trained on Run 16 (middle) and Run 20 (right) and their respective GT (left). 44
4.7 Training and validation losses for models validated on fold 2. The curves on the left are from the model trained on Run 17 with four patches, while the middle curves are from the model trained on Run 21, with seven patches. The curves on the right are from the model trained on Run 23, with thirteen patches. 45
4.8 Segmentation errors in Cirrus B-scans. The GT fluid masks, seen on the left, show a large region of PED fluid. Meanwhile, the predictions made by the models trained in Run 21 and 23, which respectively correspond to the B-scans in the middle and right, classify the center of this region as IRF. 46
4.9 Transformations applied to a vertical patch of a Cirrus B-scan. The original vertical patch (left) can be rotated a maximum of 10°, a transformation that is shown in the middle image. The image on the right represents the combination of a 10° rotation and an horizontal flip. It is seen that the rotation transform pads a significant portion of the image, reducing the information in the input. 46
4.10 Example of two B-scans in which the retina is oriented differently. The retina present in the right image is significantly more inclined. Both B-scans were obtained using a Cirrus device and do not present fluid. 48
4.11 Segmentation errors in Cirrus B-scans when trained without random rotations, in the same B-scan as in Figure 4.8. In the left, the GT masks are shown, while in the right the predicted masks are exhibited. 48
4.12 Predictions by the models trained using 5° (middle image) and 10° (right image) rotations, compared with the respective GT (left image). 49
4.13 Predictions by the model trained on Run 32 in unseen Cirrus volumes of fold 1. . 52
4.14 Predictions by the model trained on Run 32 in unseen Spectralis volumes of fold 1. 52

4.15 Predictions by the model trained on Run 32 in unseen Topcon volumes of fold 1. 53

List of Tables

3.1 Volumes, B-scans per volume, the total number of B-scans, and macular diseases in each dataset. 19
3.2 Number of OCT volumes per vendor in each fold, considering 5-fold validation. . 20
3.3 Number of OCT volumes per device in each fold, in the four remaining folds. . . 21
3.4 Layers that compose the generator and the discriminator. Each convolution is represented by Conv2d(K, OC, S), where K is the kernel size, OC is the number of output channels, and S is the stride. The same notation is used in deconvolutions, represented by TransposedConv2d. The output size is shown following C $\times$ H $\times$ W notation, where C is the number of channels, H is the height, and W is the width. The inputs have shape $1 \times 64 \times 64$. Adapted from Tran and Yang [58]. . . 33

4.1 Dice scores for every vendor and fluid for the runs done in Experiment 1.1. The conditions were the same for both sets except the extracted patches that are different in every run due to the random process of extracting them. 38
4.2 Dice scores for every vendor and fluid for the runs done in Experiment 1.2. . . . 40
4.3 Dice scores for every vendor and fluid for the runs done using four vertical patches extracted from each B-scan. In "Set 4", the models were trained in 100 epochs, while in "Set 5" the models were trained on up to 200 epochs with a 100 epoch patience. The transformations applied in these sets were the same: horizontal flipping and a maximum rotation of 10°. 42
4.4 Dice scores for every vendor and fluid for the runs done using seven (Runs 21 and 22) and thirteen (Runs 23 and 24) vertical patches extracted from each B-scan. Only two folds were used in order to understand the viability of using these numbers of patches, which is represented in the column "P". Runs 17 and 18 are shown here to make the comparison between the number of patches easier. The values shown in bold represent the best value for the models trained with the same number of patches. 44
4.5 Dice scores for every vendor and fluid using seven vertical patches. The only transformation performed in these runs was horizontal flipping, without rotation. Runs 21 and 22 are also shown, promoting an easier comparison. 47
4.6 Dice scores for every vendor and fluid using seven vertical patches. In these runs, horizontal flipping and 5° rotation were used, instead of the usual 10°. The results obtained in Runs 21 and 22 are also shown, enabling a smoother comparison. . . 48
4.7 Dice scores for every vendor and fluid using seven vertical patches. The mean scores are calculated only for the B-scans that contain that fluid. For example, Cirrus IRF is the mean of the IRF Dice coefficient across all the Cirrus slices in the validation fold that contain IRF. The transformations utilized were the same as in Table 4.6. 49

4.8 Dice scores for every vendor and fluid using four vertical patches. The rotation applied in these runs was of 5°, instead of the previously used 10°. "Set 7" and the best performing run in this set (Run 32) are also shown, promoting an easier comparison between using four and seven patches. The underlined values correspond to the best performances between the models trained in Run 32 and Run 33. . . . 50
4.9 Dice scores for every vendor and fluid in the reserved fold (fold 1). The first row reports the mean Dice across all slices. The second row shows the mean for slices containing the fluid specified in the column, while the third shows the mean for the slices without that fluid. . . . 51
4.10 IRF Dice scores for every vendor. Runs 37 to 40 utilize the multi-class 5-fold split from Experiment 1, while the Runs 41 to 44 use the fluid-specific 5-fold split. The results are presented both for every slice and for the slices which contain IRF. . . 54
4.11 SRF Dice scores for every vendor. Runs 45 to 48 utilize the multi-class 5-fold split from Experiment 1, while the Runs 49 to 52 use the fluid-specific 5-fold split. The results are presented both for every slice and for the slices which present SRF. . . 55
4.12 PED Dice scores for every vendor. Runs 53 to 56 utilize the multi-class 5-fold split from Experiment 1, while the Runs 57 to 60 use the fluid-specific 5-fold split. The results are presented both for every slice and for the slices which present PED. . . 56
4.13 Dice scores for every vendor and fluid in the reserved fold (fold 1), using the models from Runs 42, for IRF, 52, for SRF, and 59, for PED. The first block corresponds to the results when using the priority rule merging strategy, while the second block is associated with the results when using highest probability as the merging strategy. In each block, the first row reports the mean Dice across all slices. The second row shows the mean for slices containing the fluid specified in the column, while the third shows the mean for the slices without that fluid. The values marked in bold are the best values when corresponding rows are compared, so that, for example, the row "All" in the first block is compared to the row of same name in the second. . . . 58
4.14 IRF Dice scores for every vendor, using the binary segmentation U-Net for IRF. The loss function used in Runs 61 and 62 is the weighted cross-entropy. This model was trained on two different validation folds: validation fold 0 from the multi-class 5-fold split, and validation fold 4 from the 5-fold split specific for IRF. These results can be compared, respectively, to those obtained in Runs 40 and 43. The metrics are compared between rows with the same name, so that the values in bold are the best across the four experiments. . . . 60

A.1 Dice scores for every vendor and fluid for multiple runs performed in the same conditions. "Set 1" resumes Run 1 to Run 5, all of which validated using fold 2. Meanwhile, "Set 2" resumes the Runs from 6 to 14, validated in fold 3. All the runs in each set were trained on the same conditions, for 100 epochs. The random transformations applied were horizontal flipping and maximum rotation of 10° applied to it. This provides an insight of how the randomness of the U-Net affects the segmentation performance. . . . 69

List of Acronyms

AMD	Age-related macular degeneration
AMI	Anisotropic meta interpolation
BCE	Binary cross-entropy
BM	Brunch's membrane
CDFI	Compression-driven network design for frame interpolation
CNN	Convolutional neural network
CS	Contrast sensitivity
CT	Computed tomography
DME	Diabetic macular edema
GAN	Generative adversarial network
GDL	Gradient difference loss
GT	Ground truth
HR	High-resolution
ILM	Internal limiting membrane
IRF	Intraretinal fluid
MAE	Mean absolute error
MRI	Magnetic resonance imaging
MSE	Mean square error
MS-SSIM	Multi-scale structural similarity index measure
LR	Low-resolution
OCT	Optical coherence tomography
ONL	Outer nuclear layer
PED	Pigment epithelial detachment
PSNR	Peak signal-to-noise ratio
RIFE	Real-time interpolation flow estimation
RGB	Red, green, and blue
ROI	Region of interest
RPE	Retinal pigment epithelium
RVO	Retinal vein occlusion
SR	Super-resolution
SRF	Subretinal fluid
SSIM	Structural similarity index measure

Chapter 1

Introduction

The vision is the human's most important and complex sense, playing a critical role in our orientation in the world [1]. However, the health of the retina, an important part of the eye, can be compromised by multiple diseases, that lead to fluid accumulation in it. The characterization of the fluid present in the retina is important to assess the progression of diseases such as age-related macular degeneration (AMD), diabetic macular edema (DME), and macular edema secondary to retinal vein occlusion (RVO) [2].

AMD affects the macular region of the retina, leading, in later stages, to a significant and permanent loss of central visual acuity, which has a severe impact on the patient's quality of life. In patients with AMD, the formation of new blood vessels can occur, which leak fluid, lipids, and blood into the retina, resulting in the formation of retinal fluid [3]. It is one of the leading causes of visual impairment with an expected effect on 300 million people by 2040 [4].

In patients with diabetes mellitus, DME represents the most common cause of visual impairment, affecting approximately 150 million people worldwide, as of 2015. It is anticipated that this number will increase as the prevalence of diabetes in developed countries is growing [5]. The fluid accumulation is caused by a disruption of the blood-retinal barrier, which allows fluid to accumulate in the intraretinal layers of the macula, resulting in retinal thickening (edema) [6, 7].

Affecting 16 million people worldwide, RVO represents a significant cause of vision loss in older individuals. The occlusion of the retinal vein can result in swelling of the optic disc, which leads to a reduction in visual acuity [8].

The presence of intraretinal fluid (IRF) is a defining criterion of DME and RVO, while two in every three patients with AMD present this type of fluid. The majority of patients with AMD and 30% of the patients with DME and RVO have subretinal fluid (SRF). Pigment epithelial detachments (PED) occurs more frequently in patients with AMD [2].

Therefore, retinal fluids are important for the classification and progression of these diseases, and can be observed through retinal optical coherence tomography (OCT) [2]. OCT is a non-invasive imaging technique that analyzes the light behavior (such as its reflection, absorption, and time-of-flight) to estimate the spatial dimensions of the tissue's structure [9]. This allows for *in vivo* visualization of the individual retinal layers within the posterior segment of the eye. An

OCT is composed of multiple consecutive cross-sectional 2D images that, when stacked, form a volumetric representation of the posterior segment. Each of these two-dimensional images is called a B-scan. In Figure 1.1, these concepts are illustrated, showing a B-scan and the three-dimensional representation of the posterior segment of the eye. The OCT resolution is sufficiently high to assess the tissue integrity, the retinal layers, and the fluids present [10, 11]. There are multiple devices used for the acquisition of OCT volumes, resulting in different image attributes across the same technique, such as inter-slice distance, image quality, and appearance [2].

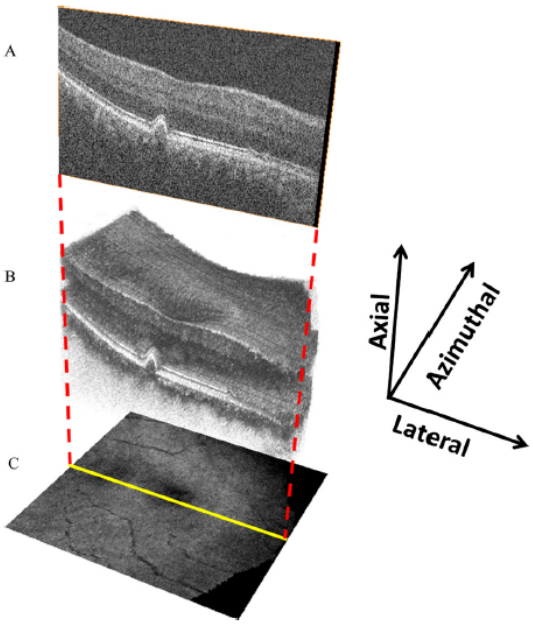

Figure 1.1: OCT B-scans (A), when taken at a fixed distance along the azimuthal axis, form a volume of the posterior segment of the eye (B) [12].

The classification of the fluid is dependent on its location within the retina. There are three different categories: IRF, which is situated in the inner and outer layers of the retina; SRF, positioned between the outer nuclear layer (ONL) and the retinal pigment epithelium (RPE); and PED, which appear beneath the RPE [2]. Figure 1.2 shows the characteristics and positions of these fluids on an OCT B-scan and Figure 1.3 exhibits the retinal layers in the OCT scan of a healthy patient.

By segmenting the fluids detected in the B-scans, their volume can be estimated and used as a progression marker of the mentioned retinal diseases. However, manual segmentation is laborious, expensive, and prone to bias, which motivates the search for automatic methods [11].

In OCT imaging, the precision of the estimated volume is not only dependent on the quality of the segmentation, but also on the inter-slice distance [14]. It is seen in other imaging techniques that the performance of the segmentation is improved when the neighboring slices are used as input. Consequently, the reduction in inter-slice distance and improvement of the resolution along this axis, betters the performance of models that include information from adjacent slices [15]. Given that the inter-slice space is reduced, the estimated segmented volume will also be closer to the real fluid volume.

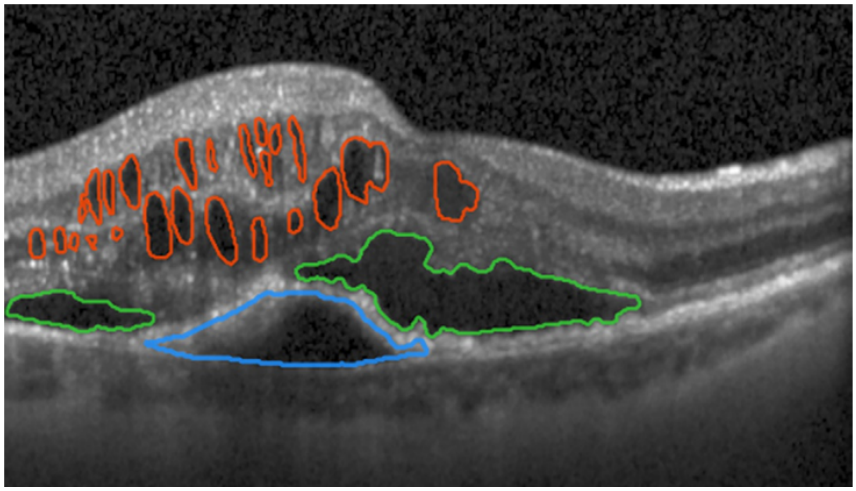

Figure 1.2: The three distinct fluid types on an OCT B-scan: IRF in red, SRF in green, and PED in blue [2].

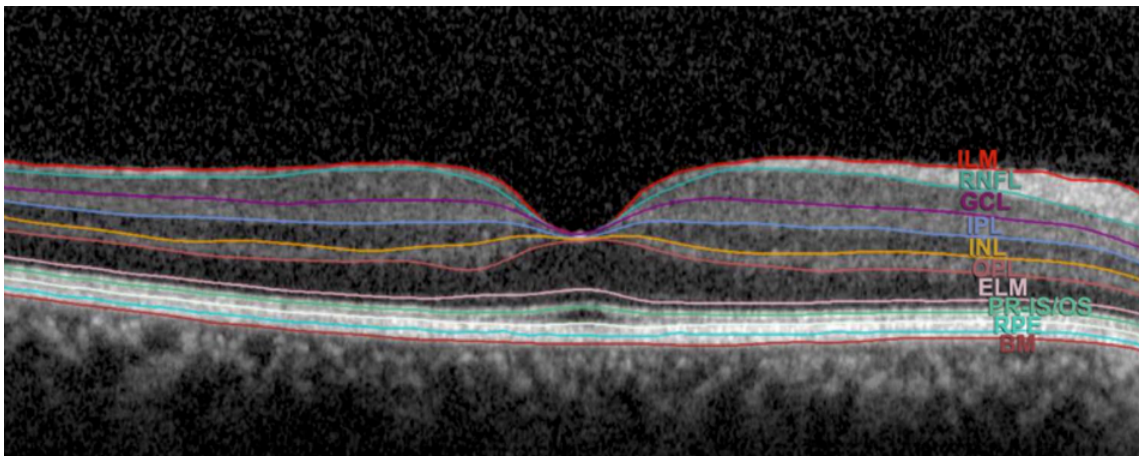

Figure 1.3: OCT scan of the retinal layers [13].

BM, Bruch's membrane; ELM, external limiting membrane; GCL, ganglion cell layer; ILM, internal limiting membrane; INL, inner nuclear layer; IPL, inner plexiform layer; ONL, outer nuclear layer; OPL, outer plexiform layer; PR-IS/OS, photoreceptor inner segment/outer segment; RNFL, retinal nerve fibre layer; RPE, retinal pigment epithelium.

Considering the previous statements, the dissertation general objective is to conduct an analysis of retinal OCT scans, classifying the retinal fluids in three distinct types (IRF, SRF, and PED) and quantifying their respective volumes. Another important objective is to increase the inter-slice resolution of the OCT volumes, with the aim of improving the fluid volume estimation. The specific objectives were determined as follows:

1. Develop different 2D deep learning models for multi-class segmentation of retinal fluids (IRF, SRF, and PED) in OCT volumes.

2. Evaluate the performance of the best segmentation model and estimate the volume of each fluid using the masks predicted by it.

3. Use a generative model for synthesizing intermediate slices in OCT volumes, generating one slice between two real slices in order to improve the inter-slice resolution of the volume, while assessing the quality of these generated images.

4. Investigate the impact of intermediate slices synthesis on the fluid volume estimation by the segmentation models.

Apart from the "Introduction", the dissertation is composed of the following chapters: "Literature Review", "Methods", "Results and Discussion", and "Conclusion". In the "Literature Review" chapter, an analysis is performed on the latest papers in the field of retinal fluid segmentation using 2D deep learning networks, as well as the latest publications on inter-slice resolution enhancement. The "Methods" chapter details the selection of the dataset for the experiments performed during the dissertation, alongside with an insightful description of the experiments performed on fluid segmentation, intermediate slice synthesis, and fluid volume estimation. In the "Results and Discussion" chapter, the results from each experiment are shared, showing the performance of each model in their respective task, while explaining the performance differences between experiments and the relationships between the variables changed and the obtained results, comparing them to the literature. Finally, the "Conclusion" shows the main findings from the experiments performed and suggests directions for further research, while exposing some limitations of the study.

Chapter 2

Literature Review

For this dissertation, research was conducted to find the most recent trends in 2D fluid segmentation of OCT volumes using deep learning and in the use of generative models in the intermediate slice synthesis.

2.1 Fluid Segmentation

In the fluid segmentation state-of-the-art research, articles were retrieved using the methodology of a systematic review. The next subsection details the retrieval process and the criteria for inclusion and exclusion of the articles. "2.1.2 Literature Review" shows the trends on the methodologies utilized for fluid segmentation.

2.1.1 Search Strategy

The search query was defined as: ""OCT" AND "segmentation" AND ("deep learning" OR "CNN" OR "neural network")". Using the query, papers were retrieved from four different databases: 398 articles from PubMed, 105 from IEEE, 125 from ScienceDirect, and 80 from ACM.

In the process of collecting the papers, those published over the previous five years and regarding 2D or 2.5D fluid segmentation in OCT volumes were included. Additionally, conferences proceedings, articles not written in English, and articles for which the full text was not accessible were excluded.

A total of 708 articles were initially identified, of which 133 were duplicates. Afterwards, 575 articles were subjected to screening, based on their titles and abstracts. These articles were analyzed in accordance with the inclusion and exclusion criteria, resulting in the removal of 499 papers. Of the remaining 76 articles for the full-text screening, 20 met the established criteria. These final articles represent the state-of-the art in 2D deep learning fluid segmentation in OCT volumes included in this dissertation.

2.1.2 Literature Review

The selected papers can be divided into two broad groups, according to the type of segmentation: binary segmentation [16, 17, 18, 19, 20, 21], where the fluid is classified in one whole class, and multi-class [22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35], where the segmented fluid is classified in two or more classes (namely IRF, SRF, and PED). We have also considered other criteria to group the papers, such as the segmentation architecture, and the use of retinal delimitation, as shown in Figure 2.1.

In binary segmentation, the approaches to the segmentation problem are simpler, but include both convolutional neural networks (CNN) [17, 18, 19, 20, 21] and transformer solutions [16]. The CNN solutions differ among them, depending on the modules that constitute each network, but all are inspired by the U-Net [36]. In Figure 2.2, an instance of a CNN used for binary fluid segmentation is shown.

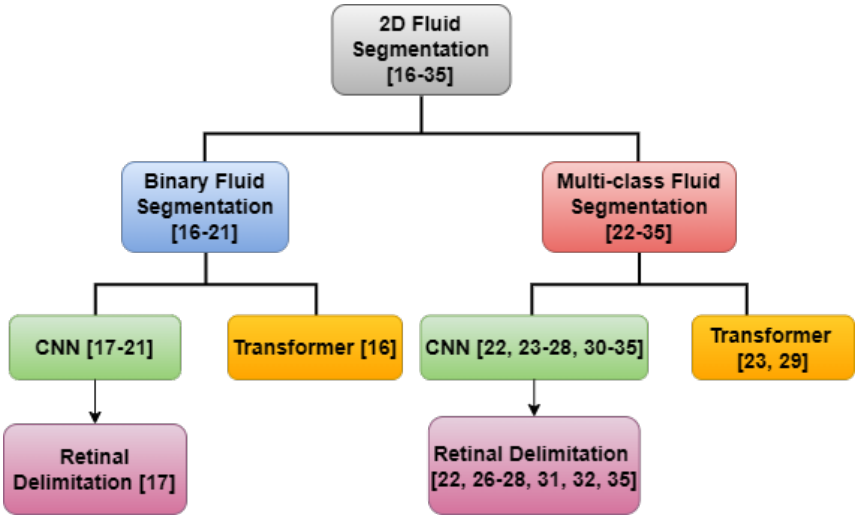

Figure 2.1: Grouping of the articles included in the literature review.

Pawan et al. [17] is the only paper in binary segmentation that restricts the input of the segmentation CNN to the content within the retinal layer. This approach is frequently observed in the papers focused on multi-class segmentation. In this article, this is achieved by performing a retinal layer segmentation and assigning all the values outside the boundaries to zero. The result of this operation is an input for the segmentation CNN. The removal of irrelevant information surrounding the retina simplifies the learning process and improves the model's focus on essential information [35].

In the framework proposed by Liu et al. [18], the slice's fluid mask and distance map are generated. The distance map consists of the predicted distance of each pixel to the background or retinal tissue, with only the values above a specified distance threshold being kept. This is achieved through the use of a double-branched network, where the encoder is the same, while the decoders vary. One decoder is responsible for generating the fluid segmentation map, while the

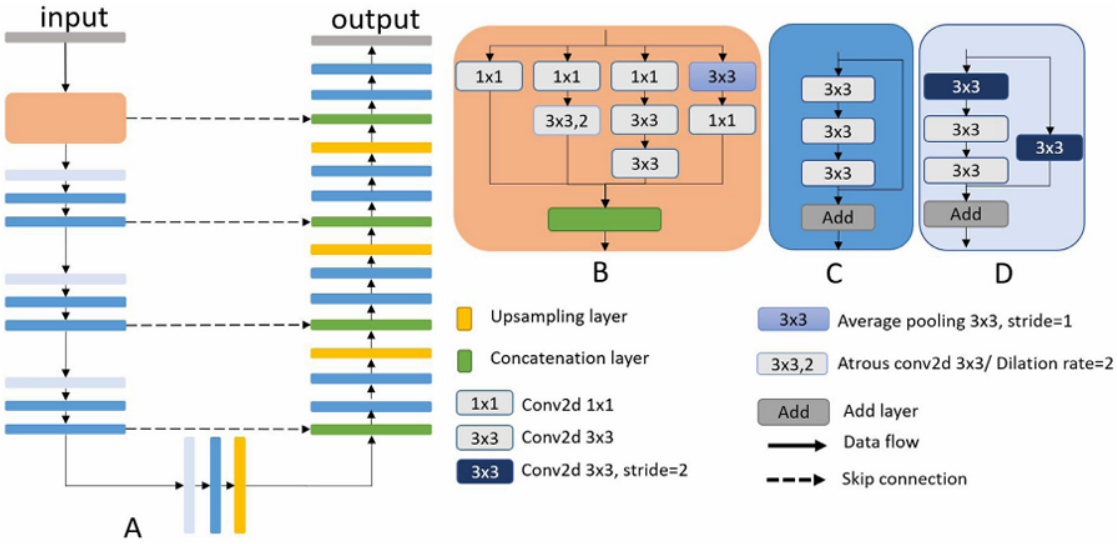

Figure 2.2: Example of a CNN architecture used in fluid binary segmentation. Image A depicts the neural network architecture. B shows the used multi-scale block, while C and D exhibit the residual convolutional blocks [19].

other predicts the distance map. The intersection between these outputs forms the final segmentation. This approach mitigates the issue of inappropriate merging of small and proximate fluid regions, as the distance map branch is better than the fluid segmentation network in discerning the boundaries that delineate fluid regions.

Resorting to generative adversarial networks (GANs), Wu et al. [21] make images from different vendors, visually similar to the images of a singular, specific vendor. Subsequently, a U-Net, which has extensively been trained on images from the specific vendor, is used for segmentation. This approach is intended to reduce the burden of learning the segmentation on multiple vendors by ensuring that all volumes are similar to one in which the segmentation model performs well. Similarly, the multi-class segmentation framework proposed by Li et al. [24] was designed based on the same idea.

CNNs inspired by the U-Net can also be combined with transformers in the context of image segmentation. While CNNs capture the information from local receptive fields, visual transformers integrate features from global receptive fields. Despite being more prevalent in multi-class segmentation frameworks, in this paper by Quek et al. [16], the visual transformers are located between the encoder and decoder paths, thus incorporating features from both receptive fields in the encoding branch.

The majority of the papers included in this review perform multi-class segmentation, therefore presenting more diverse implementations. While all these articles segment two or more fluids, Hassan et al. [28] and Padilla-Pantoja et al. [33] also segment other biomarkers. Similarly to binary segmentation, the multi-class segmentation papers can also be divided according to the presence [23, 29] or absence [22, 24, 25, 26, 27, 28, 30, 31, 32, 33, 34, 35] of transformers in the segmentation network. All the papers that have transformers in their framework, combine them

with CNNs.

Similar to what was developed in Quek et al. [16], Liu et al. [23] have integrated transformers in the bottleneck section of a segmentation network inspired by the U-Net. Liu et al. [23] utilize two networks for the segmentation: one for coarse segmentation and other for the refinement of the results from the first. Both networks are similar to the U-Net, but in the refine branch, a transformer is included. Its purpose is to provide features from global fields, compensating for the deep features that are used as input in this branch. In contrast, Zhang et al. [29] replaced the CNN encoder with a transformer encoder, exploiting its modeling capacity with self-attention.

The limitation of the input to the region within the retinal layer, ignoring what is outside of it, is seen in many of the multi-class papers [22, 26, 27, 28, 31, 32, 35], similarly to what was done in Pawan et al. [17]. There are various approaches for this delimitation, with some using CNNs trained for the segmentation of the retinal layers or the retina [32, 35], and others using algorithms leveraging on the noticeable transition between the retinal layers and its background [17, 22, 26, 27, 28, 31].

The retinal delimitation is conducted as a separate process from the fluid segmentation. In [22, 26, 27, 28, 31, 32], the retinal layer is segmented prior to the fluid segmentation, conditioning the input of the fluid segmentation network and simplifying the learning process. However, the retinal delimitation can also limit the final segmentation by intersecting the network's output, limiting the segmentation results to the boundaries of the retinal layer, as observed in Mantel et al. [35].

The fluid segmentation network input is conditioned in multiple ways. In Xing et al. [31], the image is cropped to fit its region of interest. [22, 27, 32] combined the B-scan with the retinal delimitation result, either through concatenation or along another channel. In [17, 26, 28] the information outside the retinal layer is set to zero and ignored.

Contrasting with the work of Liu et al. [18] who used a CNN to output a distance map (relative to the background or the retinal tissue), Tang et al. [32] and Rahil et al. [22], inspired by the work of Lu et al. [27], calculate a relative distance map to combine with the input slice in a CNN. Starting with the retinal delimitation, the relative distance to the internal limiting membrane (ILM) is calculated for each pixel located between the ILM and the Bruch's membrane (BM) (see Figure 1.3). This map provides information about the relative position of each pixel to the ILM, influencing their classification. An example of such framework can be seen in Figure 2.3.

Regarding the segmentation CNNs adopted by the analyzed papers, most are directly inspired by the U-Net, to which changes are done when considering the objectives of each study. Examples of such changes are the introduction of blocks (such as residual [23, 26, 28, 29, 33, 35]), and modules (like atrous sampling pyramid pooling [26, 28, 30, 34]), which makes the network distinctive. However, some papers use other variations of the U-Net that are also popular: the Deeplab [37] in Li et al. [24] and Hassan et al. [28], and the VGG [38] in Hassan et al. [26] and Padilla-Pantoja et al. [33].

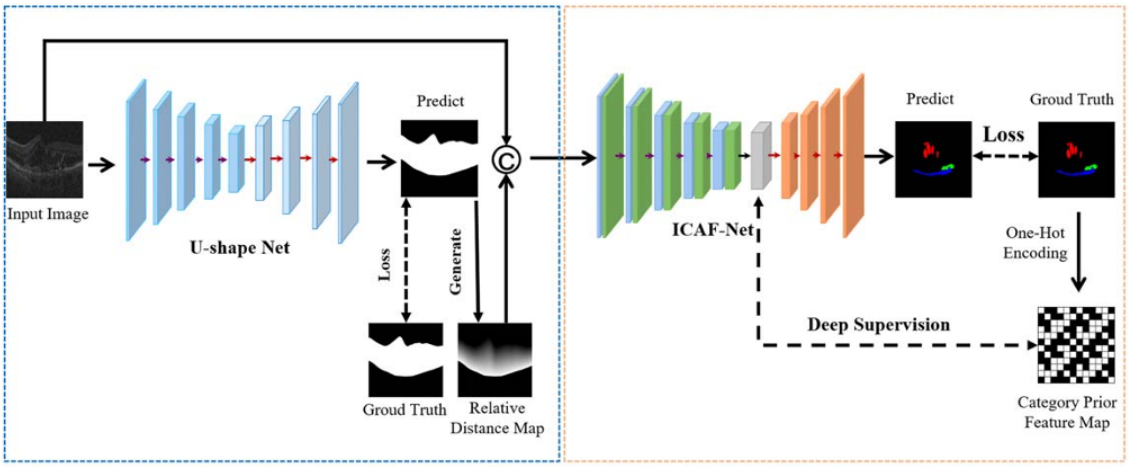

Figure 2.3: Example of a framework that includes delimitation of the retinal layer and a relative distance map (left side). The generated map is included in the segmentation network (denominated ICAF-Net, by the authors) [32].

2.2 Intermediate Slice Synthesis

For many years, there have been attempts to improve the resolution of OCT exams using computational methods, a process called super-resolution (SR). In 3D applications, such as magnetic resonance imaging (MRI), computed tomography (CT), and OCT, SR can be done intra-slice, which improves the resolution of each slice in the volume along one plane, or inter-slice, bettering the resolution of the volume along one axis, by generating one or more slices between a pair of original ones. Some frameworks may even contain both approaches [39].

The use of GANs to generate slices between other known slices is commonly used technique in MRI and CT, but with few examples in OCT [39]. The systematic literature review performed by Ibrahim et al. [40], which analyzed the latest trends in the use of generative models in medical data, only presents one example of GAN for inter-slice resolution improvement in OCT volumes [14]. In this imaging technique, the use of GANs is mainly done for the generation of OCT images and conversion between different vendors [40].

In the following subsection, the state-of-the-art architectures used for the improvement of inter-slice resolution are presented.

2.2.1 Architectures

Given the lack of examples in OCT imaging, it was considered appropriate to study works from other imaging techniques, such as CT, MRI, and even video, given that the working principle is the same across them. The selected papers that are applied to medical images can be classified into three distinct categories: inter-slice SR, which leverages information from adjacent slices to generate one or more intermediate slices [14, 41, 42, 43]; intra-slice SR combined with inter-slice SR, which improves the resolution of the slices from orthogonal planes and combines them with

the results of inter-slice SR [44, 45, 46, 47, 48]; and SR applied directly in 3D volumes, utilizing three-dimensional convolutions in the generation process, which incorporates the information along all the axes from multiple slices simultaneously [49, 50, 51, 52].

López-Varela et al. [14] present an inter-slice SR framework based on a GAN, inspired by the ResNet, for the generation of three B-scan slices between two known slices. The GAN training process, as illustrated in Figure 2.4, begins with the generation of an intermediate slice (Central Fake) located between two original B-scans (Pre and Post), which are separated by another original one (Central). The Central B-scan serves as the ground truth (GT) and is used in the assessment of image quality generated by the network. Subsequently, the network generates other two slices: one between the Pre and Central slices, designated as Pre-Central fake, and another between the Central and Post slices, named Post-Central fake. As the mentioned generations lack a corresponding GT, the network performance is regulated by using these two new synthetic slices to generate an additional Central Fake (Central Fake 2), which is then compared to the true Central. Consequently, if the generation of Pre- and Post-Central fakes are inadequate, the Central Fake 2 will also be of poor quality, resulting in a higher loss value. During the inference process, one slice is synthesized for every two known B-scans, reducing the inter-slice distance to half of the original value.

The importance of this study comes not only from it being the only study in OCT but also from the approach selected, which is similar to the foundation of the frameworks implemented in other papers.

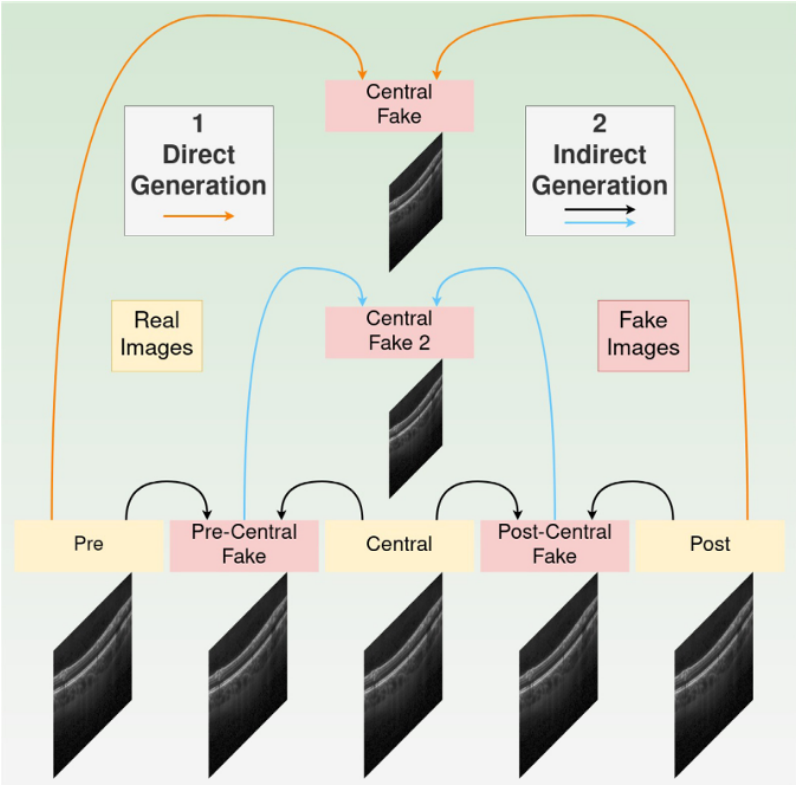

Figure 2.4: López-Varela et al. [14] training process.

In a more straightforward approach, Nishimoto et al. [43] utilize a baseline U-Net that uses two spaced slices as input to generate the slices between them. This methodology was tested in the generation of three, four, and five intermediate slices, and obtained better outcomes than those generated through linear interpolation. This approach works particularly well due to the low noise present in the CT scans. In volumes with more noise, due to the presence of metal artifacts, the slices generated using the U-Net are of worse quality than those obtained through linear interpolation.

The work by Xia et al. [41] demonstrates the enhancement of inter-slice resolution in MRI, through the utilization of multiple networks and a multi-scale discriminator that considers both the image from a large and a small field of view images. Therefore, the networks of this framework receive two consecutive slices ($x_{z\pm1}$) and attempt to generate one between them (x_z). The generator (G) in this framework attempts to generate the intermediate slice, while the discriminator (D) learns to distinguish the synthesized image from the real image. The loss of these networks is also determined by the similarity of the feature maps for different images (L_{FM}).

The framework contains two other U-shaped networks: one that learns to predict the optical flow and one that learns to predict the depth map for each of the two input images. Parting from the optical flow and depth map of the input images and their linear interpolation ($\bar{x_z}$), another U-shaped network generates the intermediate slice. The result from this network and the image output from the generator, to which is applied Gaussian blurring, are evaluated by the discriminator. Once again, the discriminator evaluates both images in different feature maps, comparing them.

The image generated by the network that uses optical flows and depth ($\hat{x_z}$) has special attention to the transitions between the two input slices. The point of inputting these images to the discriminator is to incentive it to look for these characteristics in the images output from the generator, a network which has a larger capability of reproducing outputs more visually similar to real images. Similarly, the discriminator also receives blurred images to incentive the generator to output sharper images. A scheme of this framework can be seen in Figure 2.5.

Similarly, Wu et al. [42] improved the inter-slice resolution by training a generator network to output bi-directional spatial transformations instead of producing fake images. The advantage of this process is that it allows the same transformations to be applied to the segmentation masks from the surrounding slices, generating fake masks for the fake slices.

As in Xia et al. [41], the GAN's discriminators also judges the generated images in both a larger and smaller field of view. To evaluate the output at a larger field of view, a global discriminator classifies the image in real or fake. Meanwhile, the classification at a smaller field of view integrates an attention network, which focus on the most useful parts of the image that help the local discriminator and the object identifier.

The local discriminator classifies the image as real or fake based on these smaller features output from the attention network. Meanwhile, the object classifier, which is a much shallower network, verifies if certain structures that are present in the real image also appear in the fake one, parting from the output of the attention network.

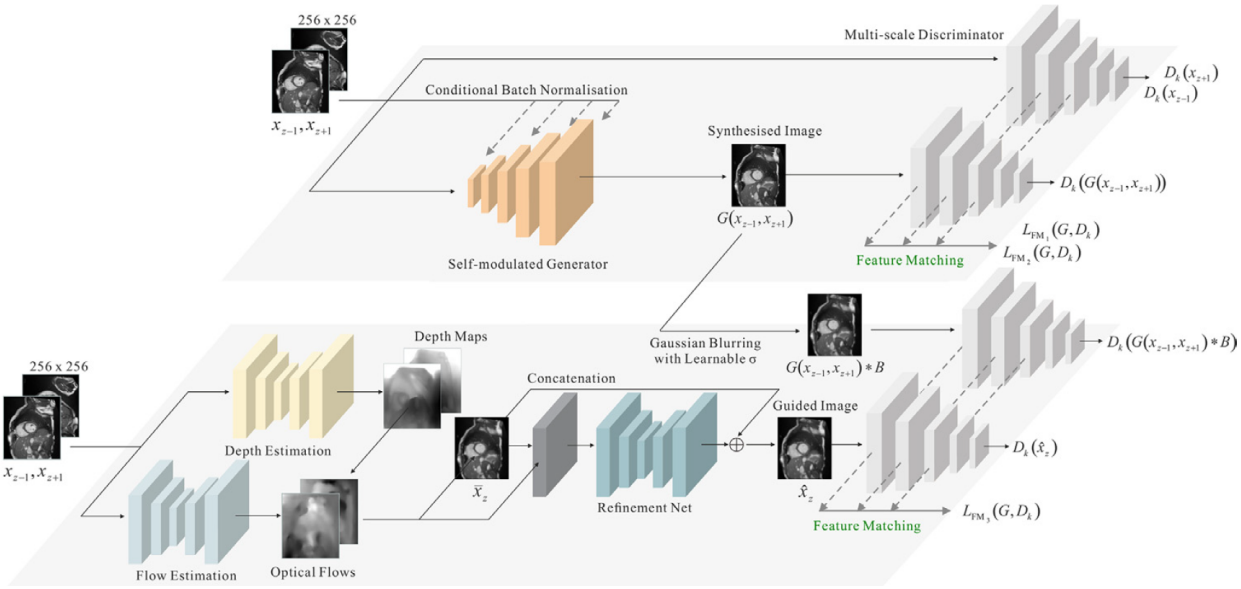

Figure 2.5: Framework developed by Xia et al. [41].

By using the global and local discriminator, the generator is encouraged to output images that closely resemble the real images, both in the overall structure and in the fine details. The object detector ensures that the objects present in the real image are represented in the generated one.

As an example of using intra-slice SR to improve inter-slice resolution, Zhang et al. [44] implemented two networks that enhance the resolution of CT volume's slices in the two planes with the lowest in-slice resolution: sagittal and coronal. These networks increase the resolution only along the axial direction. The models here utilized are based on the anisotropic meta interpolation (AMI) network developed by Peng et al. [45], a benchmark work in the field of medical image SR.

Peng et al. [45] introduced the AMI network, a single image SR network designed to enhance the slice's resolution along the axis of lowest spatial resolution, which is the axial axis, in this case. The AMI network is applied independently to the sagittal and coronal slices, producing in complementary interpolations along the axial direction. A fusion network then combines these two outputs to synthesize high-resolution slices along the axial plane.

The implementation by Zhang et al. [44] extends the use of the AMI network in the images of the sagittal and coronal planes by incorporating a GAN. This GAN receives two adjacent axial slices as input and attempts to generate the intermediate slice, with a framework that is similar to that of López-Varela et al. [14].

The three networks are trained on downsampled CT scans from which every other axial slice is removed. The outputs of the sagittal and coronal AMI networks are compared to the corresponding slices in the original CT scans. Meanwhile, the GAN's output is compared to the GT axial slices from the original CT scan.

Instead of using a dedicated fusion network as in Peng et al. [45], Zhang et al. [44] introduce a loss function that directly compares the outputs of the AMI networks to the output of the GAN. This loss is backpropagated through the generator, encouraging it to output axial slices that are

coherent with the content inferred by the AMI networks. This results in axial images that are not only visually consistent with the original CT slices but also integrate structural information from other anatomical planes. In Figure 2.6, a scheme of this method is shown.

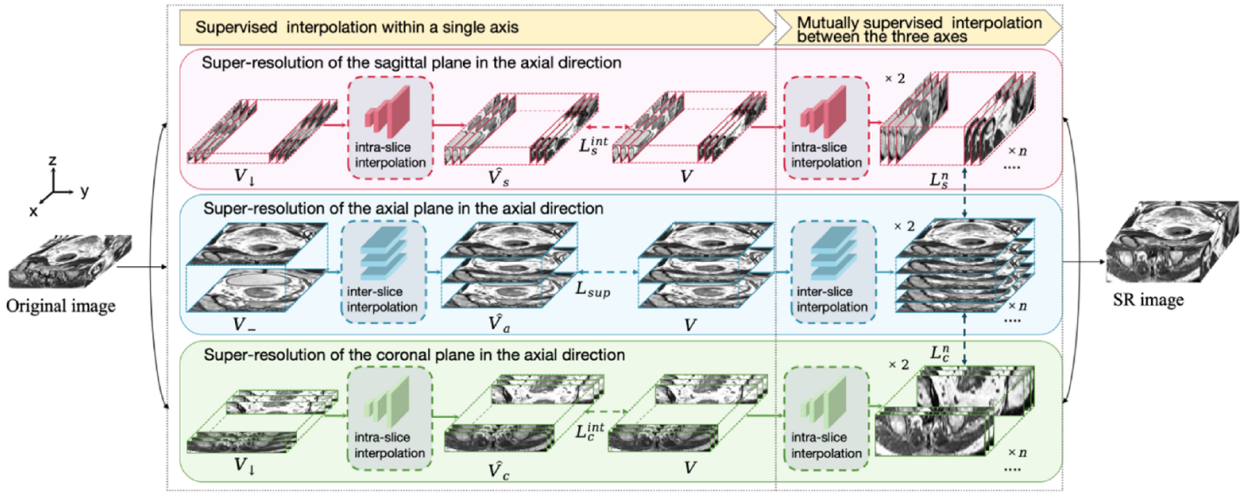

Figure 2.6: Architecture of the method developed by Zhang et al. [44].

A similar approach was done by Fang et al. [46]. In this framework, three CNNs (one for each axis) are trained with the goal of generating the intermediate slices along the axis with a lower inter-slice resolution, which is the axial axis. Similarly to what is done in Zhang et al. [44], two CNNs (one for the sagittal plane and one for the coronal plane) are trained to increase the resolution along the axial axis in the images of their respective plane. Both CNNs used in this work are the single image SR model developed by Niu et al. [53]. This CNN is an holistic attention network which improves the image's resolution by capturing both local and global contextual information. It uses holistic attention modules that combine channel attention, which emphasizes informative feature maps, and spatial attention, which highlights the most important regions of the image. This attention mechanism allows an image reconstruction with finer textures and sharper details in the super-resolved output.

Instead of using a GAN to generate the intermediate slices along the axial plane, the authors apply another CNN. In this CNN, they start by extracting feature maps through space-to-depth transformation operations, which reorganize the spatial information into the channel dimension. These features are then processed in a U-shaped architecture which captures both local and global context. Finally, a depth-to-space transformation is applied to the output layer, reconstructing the intermediate axial slice in the original spatial resolution.

All the networks are trained on downsampled CT volumes. After the first interpolation, all the outputs are compared to their respective GT and their loss is calculated. Then, another interpolation is performed in each axis, and a loss compares the generated images not with the GT, but with each other. This ensures coherence in the outputs predicted on different axis and transference of knowledge between them. During testing, the model inference is done through a weighted prediction with each axis' network contributing equally. In Figure 2.7, the pipeline describing this

implementation is shown.

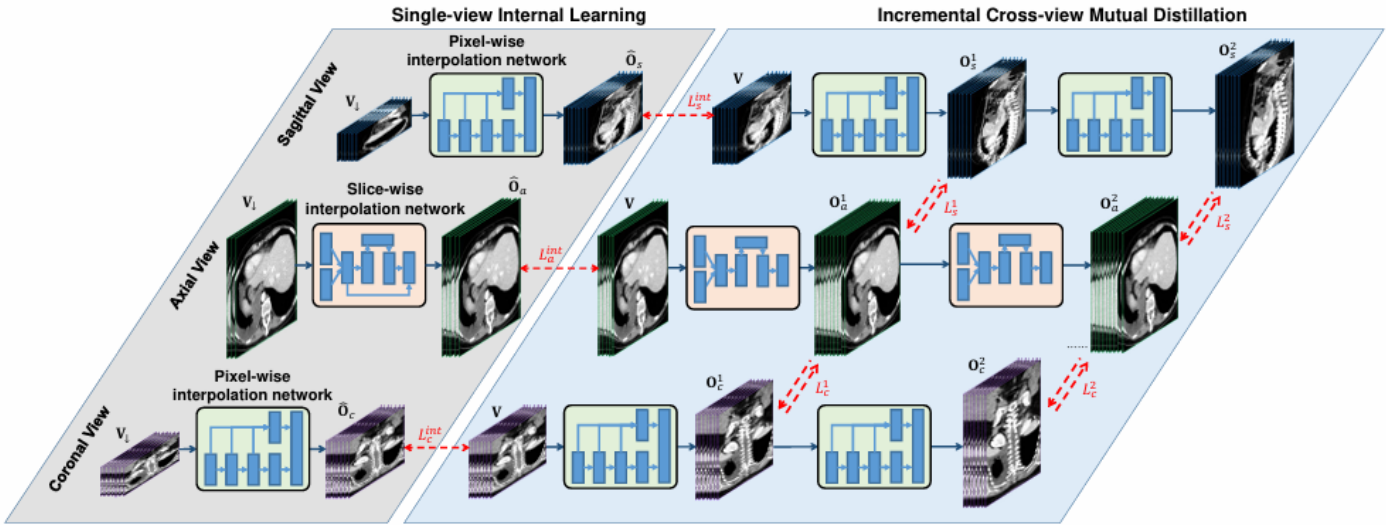

Figure 2.7: Pipeline of the methodology utilized by Fang et al. [46].

Nimitha and Ameer [47] also perform intra-slice and inter-slice resolution enhancement. However, instead of improving both resolutions simultaneously as seen in Zhang et al. [44] and Fang et al. [46], the enhancement is done sequentially. The pipeline starts with a low-resolution (LR) volume whose axial slices have their resolution enhanced, becoming high-resolution (HR). Then, parting from the HR axial slices, an intermediate slice is generated between a pair of two consecutive axial slices.

The network used in the enhancement of intra-slice resolution is a GAN. The generator starts with an in-plane and out-of-plane attention network that extracts features from the MRI slices adjacent to the one desired to enhance. After this, the generated features are input through U-shaped network that reconstructs the intermediate image. However, the number of up-sampling blocks is higher than the number of down-sampling blocks to ensure that the output has larger dimensions than the input.

To generate the intermediate slices, the compression-driven frame interpolation (CDFI) network, a state-of-the-art video frame interpolation CNN developed by Ding et al. [54], was utilized. CDFI is built on top of the AdaCoF module [55], which is particularly powerful at handling a wide range of motions between images.

The CDFI network first extracts features using a U-Net architecture. These features are passed from the U-Net encoder to a pyramid network with the same number of levels as the number of encoders in the U-Net. At each level, a 1x1 convolution is applied to the feature maps to reduce dimensionality and prepare them for motion estimation.

Two sub-networks take the U-Net's output and estimate the parameters of the AdaCoF operation. The U-Net's output is also combined with an occlusion mask to create the first candidate intermediate slice.

The pyramid network's outputs are warped by the AdaCoF and input into a synthesis network that outputs the second candidate intermediate slice. The two candidate intermediate slices are combined through a second occlusion mask to generate the final output. This model keeps visual realism output from the first slice, while ensuring a coherent pixel movement by using the second slice.

A similar approach was also used by Georgescu et al. [48] to enhance the intra- and inter-slice resolution in CT and MRI scans through two separate CNNs. One CNN was tasked with enhancing the resolution of the LR slices. Concurrently, the other CNN was utilized to reduce the distance between slices by increasing the resolution of the images from the orthogonal plane. By inferring an image with increased resolution along this plane, new intermediate slices were generated, improving the inter-slice resolution along the low resolution axis.

The CNNs for intra- and inter-slice resolution are similar. Both consist of 10 consecutive convolutional layers, with the first 5 having 32 filters. Then, the number of filters in the following layers changes according to the output size. This output size is different for the two CNNs because the first increases the resolution in two directions, while the other only increases the resolution in one direction.

The methodologies utilizing 3D GANs are similar between each other, as they all apply networks based on the GANs implemented in 2D images. As this method already considers the information across all the axes simultaneously, there is no need to use multiple networks for each, as seen in some of the previous approaches. Therefore, the differences between papers mainly originate from the medical imaging technique to which it is applied, the modules that constitute the 3D GANs used, and the datasets used for evaluation [49, 50, 51, 52].

Applications to increase the number of frames per second in a video work based on the same principle as the implementations that increase the inter-slice resolution of the three-dimensional volumes in medical images. In order to increase the number of frames in a second of the video, these frameworks utilize two consecutive frames to generate an intermediate one, which is similar to what is seen in the previous papers that perform intermediate slice generation in medical images. The key difference between these applications is that, in video, the physical quantity that separates the frames is time, while the physical quantity that separates slices in a medical imaging volumes is distance [46, 56].

Believing that the concepts that work on video also work on CT and MRI, Gambini et al. [56] implemented a state-of-the-art method of video interpolation to generate intermediate slices in CT and MRI. The method used was real-time interpolation flow estimation (RIFE) [57]. The CNN used in RIFE learns the pixel movements between frames by seeing numerous examples. This approach, called contextual flow, appears as an alternative to optical flow, a method commonly utilized to describe the movement between frames which attempts to predict pixel movements by calculating their movement between consecutive frames. RIFE is also aware of the time difference between frames, which translates to the distance between slices in CT and MRI. To construct the middle image, the network learns how to blend the previous and following image so that the generated intermediate one looks more similar to the expected, combining it with the contextual

flow. Lastly, a second network is used in the refinement of the generated image.

Both networks learn based on a loss function that has three components: a photometric loss that determines how close the generated image is to the ground truth; a perceptual loss which evaluates the generated image as the human perception would; and a smoothness loss that evaluates how smooth the generated image is [57].

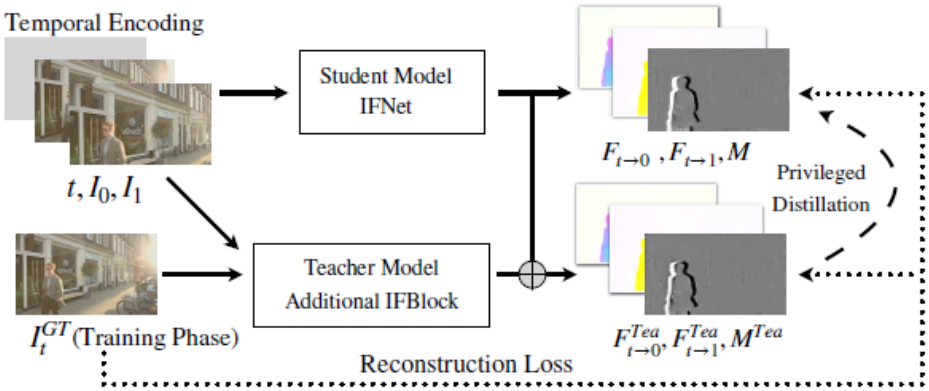

Figure 2.8: Pipeline that describes the RIFE framework. The student model attempts to generate the intermediate frame, while the teacher refines the frame generated by the student so that it looks more similar to the middle frame. The results from both networks are evaluated on the reconstruction loss [57].

Tran and Yang [58] present an alternative framework to the video frame interpolation. Instead of recurring to contextual or optical flow to understand how the pixels change between images, two GANs are used to predict the intermediate frame. The first generator receives as input the previous and following slice of the one desired to segment. The resulting slice is evaluated using the generator loss, which is composed of four components. This loss evaluates the reconstruction of the image when compared to the true image and evaluates how well it fools the discriminator.

The image resulting from the generator is input to the discriminator which is responsible for correctly classifying it as real or fake. The prediction of the discriminator is compared to the true image's label and the loss that evaluates this performance is used in the adjustment of the discriminator weights.

After training the first GAN, the second one is trained, which is a pix2pix [59]. Similar to what is seen in the work of Gambini et al. [56] and Huang et al. [57], where a second network is used in the refinement of the output of the first one, this GAN is responsible for making the image output from the first more similar to the original image. Contrasting with the previous examples where both the generative and refining networks are trained at the same time, the refining network is trained independently and after the training of the first network. The pipeline that describes this framework is shown in Figure 2.9.

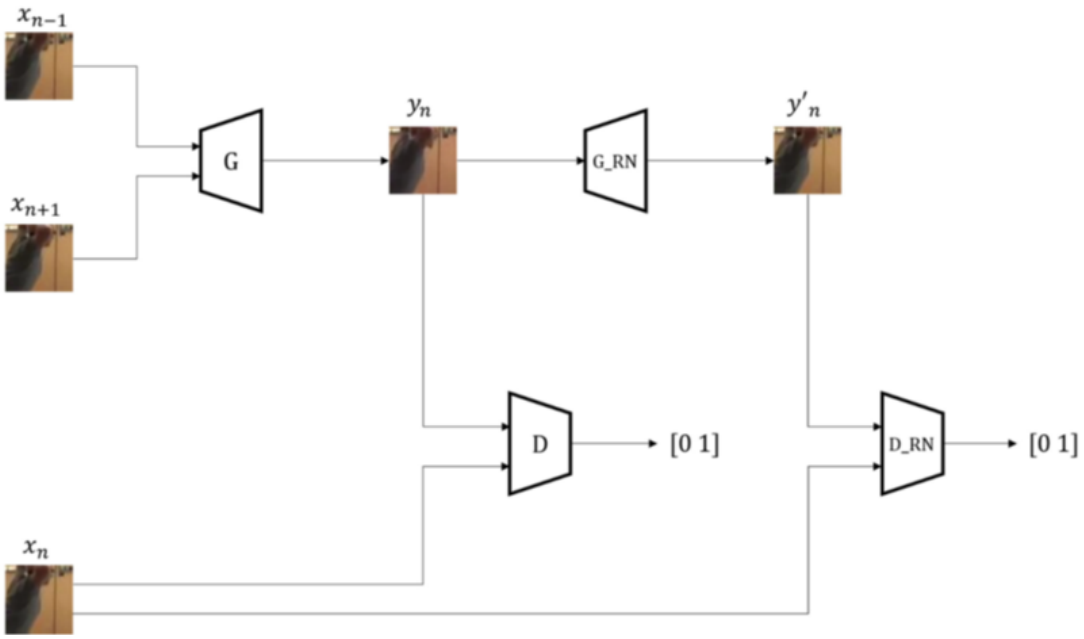

Figure 2.9: Pipeline representing the framework developed by Tran and Yang [58]. The generator that generates the intermediate frame is represented by *G*, while its discriminator is labeled *D*. The pix2pix generator is denoted by *G_RN* and the discriminator is represented by *D_RN*. x_{n-1} and x_{n+1} respectively represent the previous and following frames of the one that is being generated, x_n. y_n is the image generated by the first generator, while y'_n is the image refined by the pix2pix network [58].

Chapter 3

Methods

The Methods chapter starts with an overview of the dataset selected for the fluid segmentation and intermediate slice synthesis tasks, while regarding the requirements needed for the training of each model and the reasoning behind the selection. Afterwards, it provides an explanation of the experiments that were performed during the dissertation, regarding fluid segmentation, inter-slice generation, and fluid volume estimation, while explaining the methodologies that were implemented.

3.1 Dataset

The application of deep learning to fluid segmentation in OCT volumes requires a large number of images annotated with the three retinal fluids for the training process. The manual segmentation of large amounts of B-scans is a laborious process, which results in a shortage of publicly available annotated OCT datasets. Consequently, the majority of these datasets contain a limited quantity of images.

The dataset selected for this dissertation is the RETOUCH dataset [60]. This dataset consists of 112 OCT volumes, obtained with four different devices: 38 from the Cirrus HD-OCT (Zeiss Meditec), 38 from the Spectralis (Heidelberg Engineering), and 36 from the T-1000/T-2000 (Topcon). The 112 volumes are split into training (70 volumes) and testing (42 volumes). Only those in the training set have annotations of the retinal fluids (IRF, SRF, and PED). For the training and testing of the segmentation models, only the annotated volumes were used.

From the 70 volumes, 24 were obtained with the Cirrus, 24 volumes were acquired with the Spectralis, and 22 were obtained with the two Topcon devices. The number of B-scans per volume, the dimensions of the B-scans, and the axial resolutions vary according to the device utilized to obtain the OCT. The volumes acquired using the Cirrus have 128 B-scans, while those obtained with Spectralis have 49 B-scans. The volumes acquired using Topcon devices (T-1000 or T-2000) have 128 B-scans, but there are two volumes that only contain 64 B-scans. In total, 6838 B-scans were used on the train and test of the segmentation models.

When compared with other renown OCT datasets annotated with retinal fluid, such as the Duke dataset [61], the two datasets from the University of Minnesota [62, 63], and the Lu et al. [27] dataset, the RETOUCH presents a significantly larger quantity of annotated volumes. It also shows more variety since the volumes were obtained using four different devices instead of including volumes from just one device, as done in the mentioned datasets. In Table 3.1, a comparison between the number of annotated B-scans in each of the mentioned datasets is shown, as well as the devices utilized to obtain the OCT images, the diseases of the patients, and the distribution of annotated B-scans per OCT volume.

Table 3.1: Volumes, B-scans per volume, the total number of B-scans, and macular diseases in each dataset.

	DUKE2015 [61]	**UMN2017 [62]**	**UMN2018 [63]**	**LU2019 [27]**	**RETOUCH [60]**
Volumes	10	24	29	528	70[a]
B-scans/Volume	11	25	25	Variable	128 (Cirrus and Topcon[b]), 64 (Topcon[b]), 49 (Spectralis)
B-scans	110	600	725	750	6838
Device	Spectralis	Spectralis	Spectralis	Spectralis	Cirrus, Topcon and Spectralis
Disease	DME	AMD	DME	DME	AMD and RVO

[a] 24 volumes from Cirrus, 22 volumes from Topcon, and 24 volumes from Spectralis.
[b] Two of the training volumes obtained using the Topcon devices have only 64 slices.

For these reasons, the RETOUCH dataset is regarded as a diverse and large dataset, widely used in the literature that aims to perform fluid segmentation using deep learning, as done in [22, 23, 24, 26, 27, 29, 31, 32]. These aspects motivated the selection of the RETOUCH as the dataset that was used for implementing the models for fluid segmentation in OCT volumes in this dissertation.

In intermediate slice synthesis, the 112 OCT volumes that constitute the RETOUCH dataset were used for the training and evaluation of the models. The volumes that do not have segmentation masks can also be included since these masks are not necessary in the intermediate slice generation task.

The consistent number of slices per volume and large quantity of OCT volumes make the RETOUCH dataset suitable for the training and evaluation of the models developed to generate intermediate slices.

3.2 Experiments

In this subsection, the experiments conducted during the dissertation are explained in depth. The subsection begins with a description of how the data was split, followed by the experiments in fluid segmentation, intermediate slice generation, and in fluid volume estimation.

All the experiments were conducted using an NVIDIA GeForce RTX 3080 GPU and the PyTorch machine learning library (version 2.5.1).

3.2.1 Cross-validation

To promote consistency across all experiments, the conditions were held identical. In every experiment, the train-test split followed a 5-fold split, with different splits being used for the segmentation and generation tasks. During training, all the images in three folds were utilized to train the model, while the images from one fold were used in its validation. One fold was reserved and used to compare the performance between the best model from different experiments. Therefore, four training runs are completed in each experiment, rotating the validation fold across runs. The reserved fold consists of the same OCT volumes for all experiments, allowing for further comparisons on data not seen by any of the models.

The images in the fold that was used in validation allowed an insight of how the model was learning. In the segmentation experiments, the instance of the model that achieved the lowest loss on validation data was saved, as this typically indicates the best generalization performance on unseen data. Also, when the model was no longer improving, training could be stopped, saving computational resources.

The dataset was split so that the quantity of each fluid per vendor and the number of volumes per vendor was equally distributed across the folds. By equally distributing the volumes, it is easier to assess the model's learning capability and its behavior towards data with different characteristics (e.g. data from different vendors). To accomplish a fair data split, a custom algorithm was elaborated. This algorithm sought to divide the data in five folds while minimizing the differences of fluid per vendor and the number of slices per vendor in each fold.

A possible distribution of 70 OCT volumes from the RETOUCH dataset, which were used in the training of fluid segmentation, can be seen in Table 3.2. The split was applied to the volumes and not to the slices. The slices of the same volumes must be kept together to prevent data leakage, where similar images, obtained from the same patient, are present both in training and validation, leading to over-optimistic performance metrics.

Table 3.2: Number of OCT volumes per vendor in each fold, considering 5-fold validation.

Vendors	**1**st	**2**nd	**3**rd	**4**th	**5**th
Cirrus	5	5	5	5	4
Spectralis	5	5	5	5	4
Topcon	$4^a + 1^b$	$4^a + 1^b$	4^a	4^a	4^a

Volumes marked with ***a*** consist of 128 B-scans.

Volumes marked with ***b*** consist of 64 B-scans.

In 3.2.2.2 Experiment 2, where the model performs binary segmentation of each fluid, the volumes can be redistributed using the same algorithm, with less bounds. In this experiment, it is relevant to split the volumes in folds based only on their vendors and quantity of the fluid to segment, thus eliminating the restrictions imposed by the quantity of other two fluids. Nevertheless,

the volumes that are in the previously defined reserved fold can not be used nor in training nor in validation.

In the inter-slice generation experiments, the 5-fold split was not done by considering the fluids quantity in each fold. Since in this experiment the test volumes of the RETOUCH dataset were used and there are no fluid masks available, the quantity of fluid in each test volume is unknown. However, one of the folds in the inter-slice 5-fold split is the one reserved in the multi-class segmentation split.

The split was performed by taking into consideration solely the number of slices per device. In this experiments, the characteristic's of each device are important, since each device has a specific inter-slice distance, which is different even across devices of the same vendor and an important characteristic in image generation.

Considering both training and testing volumes of the RETOUCH dataset, there are 38 Cirrus, 38 Spectralis, 13 Topcon T-1000 and 23 Topcon T-2000 (two of which with 64 slices). The fold reserved in the multi-class segmentation task is composed of the following volumes: 4 Cirrus, 5 Spectralis, 3 Topcon T-1000, and 2 Topcon T-2000 (one of which with 64 slices). The volumes remaining for the four folds used in the generation task can be distributed as in the Table 3.3.

Table 3.3: Number of OCT volumes per device in each fold, in the four remaining folds.

Devices	**1**st	**2**nd	**3**rd	**4**th
Cirrus	9	9	8	8
Spectralis	9	8	8	8
T-1000	5	5	5	4
T-2000	3	2	2	2
T-2000^{a}	1	0	0	0

a: volumes with 64 B-scans.

Since the partition is not bounded by the quantity of fluid in each volume, it is possible to compute the best partition by iterating through all the possible combinations. In each combination, the standard deviation of the total number of B-scans in each fold is calculated. The combination with the smallest deviation was used.

Similar to what was done in the fluid segmentation task, three folds was used in training while one was used in validation. The reserved fold was used as a comparison between the best generative models from different experiments.

3.2.2 Fluid Segmentation

The initial experiments of this dissertation focused on training networks on the fluid segmentation task. The goal of these experiments is to determine which segmentation network performs the best in the considered task, which were later required for the fluid volume estimation.

In these experiments, the U-Net [36] was used in the multi-class segmentation of the fluid regions in each B-scan. The U-Net is distinguished by its encoder-decoder structure, which resembles the letter U (see Figure 3.1). In the encoder path, two 3x3 unpadded convolutions are applied to the input image, with each being followed by a rectified linear unit (ReLU) and a 2x2 max pooling operation with a stride of 2, downsampling the image. In each downsampling step, the number of channels is doubled. In the expanding path, a 2x2 up-convolution is used, resulting in the halving of the number of channels. The result is then concatenated with the cropped feature map from the respective contracting path. A 1x1 convolution is applied to the final layer.

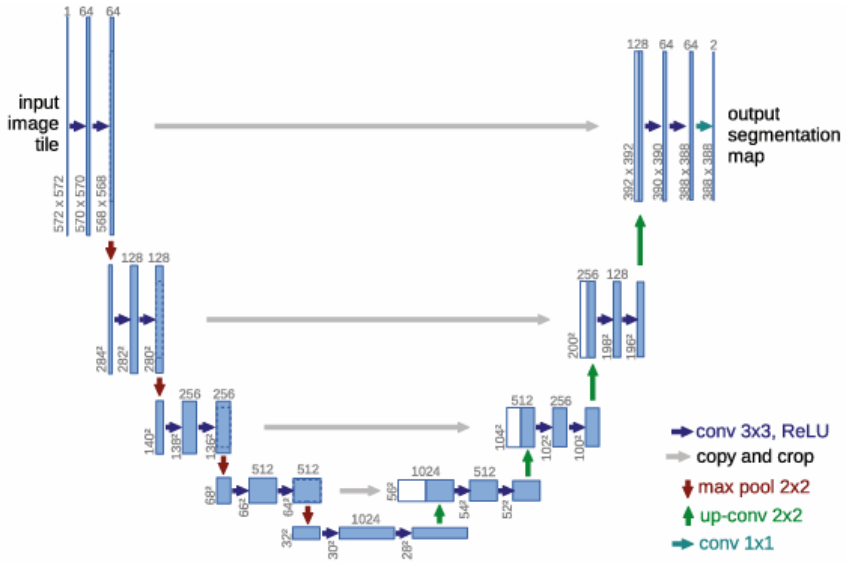

Figure 3.1: U-Net architecture [36].

The evaluation of all networks was conducted using the Dice coefficient. The Dice coefficient is a commonly used metric for evaluating the similarity between two sets. In this context, it was used for assessing the similarity between the segmentation mask generated by the segmentation network and the GT. The equation that describes the Dice coefficient can be seen in Equation 3.1, where A is a set that represents the GT binary mask of one fluid and B is another set that represents the predicted binary mask of the same fluid [64]. Considering a_i and b_i the binary pixels, the Dice coefficient can be rewritten as shown in Equation 3.2. The network that performed the best was selected to estimate the fluid volumes in the fluid volume estimation experiments.

$$\text{Dice}(A,B) = \frac{2|A \cap B|}{|A| + |B|} \tag{3.1}$$

$$\text{Dice}(A,B) = \frac{2\sum_i a_i b_i}{\sum_i a_i + \sum_i b_i} \tag{3.2}$$

The loss function that regularized the training in the fluid segmentation experiments was the same as the one used in Tennakoon et al. [65], whose segmentation model was previously implemented by the authors. This loss is described as seen in Equation 3.3, where λ_D is the weight of

the Dice component, $\mathscr{L}_D$, and λ_{CE} is the weight of the cross-entropy component, $\mathscr{L}_{CE}$, with both weights being 0.5.

$$\mathscr{L} = \lambda_D \mathscr{L}_D + \lambda_{CE} \mathscr{L}_{CE} \tag{3.3}$$

The component $\mathscr{L}_D$ is the Dice loss of the foreground. This translates to how good the model is at detecting and segmenting the fluid present in the B-scans. For any image, where each pixel is associated with an index i, the loss can be described through Equation 3.4, where $s_{i\bar{0}}$ is a binary variable that is 0 when the pixel i belongs to the class 0 (background) and is 1 whenever the pixel i belongs to any class that is not 0 (foreground). $p_{i\bar{0}}$ corresponds to the predicted probability of the pixel i belonging to the foreground. The ε constant is a small value utilized to prevent division by zero.

$$\mathscr{L}_D = 1 - \left(\frac{2 \sum_i s_{i\bar{0}} p_{i\bar{0}}}{\sum_i s_{i\bar{0}} + \sum_i p_{i\bar{0}} + \varepsilon} \right) \tag{3.4}$$

However, this loss component is not enough to correctly label the pixels in their respective classes and, for that reason, the cross-entropy component was used. Due to the large class imbalance in the images, with the background occupying the majority of them, the cross-entropy is balanced by taking into account the number of pixels belonging to each class. The cross-entropy is calculated for each pixel of index i belonging to the image. Then, for each class, the cross-entropy of all pixels in the image is summed, before being divided by the number of pixels that belong to the class. The mean of the values obtained for each class result finally in $\mathscr{L}_{CE}$, as can be seen in Equation 3.5. In this equation, $N = 4$ and is the number of classes, while C is the set of possible classes, $\{0, 1, 2, 3\}$, which corresponds, respectively, to background, IRF, SRF, and PED.

$$\mathscr{L}_{CE} = -\sum_{c \in C} \frac{1}{N} \left(\frac{1}{\sum_i s_{i,c}} \sum_i s_{i,c} \ln p_{i,c} \right) \tag{3.5}$$

3.2.2.1 Experiment 1

In the first experiment, the base U-Net model was trained to perform 2D multi-class segmentation of the retinal fluids in OCT volumes.

This was the most extensive set of experiments, where many variables were tested. Different patch shapes, transformations, and hyperparameters were experimented, until the best training settings were determined. The best settings were then used in 3.2.2.2 Experiment 2. All the experiments were done using the Adam optimizer [66] with a learning rate of 2×10^{-5}.

Experiment 1.1 The model was initially trained on patches of size 256 × 128 (H × W), following the same implementation as the one in Tennakoon et al. [65]. The extraction of patches aims at prioritizing the B-scan information relevant for the segmentation. To achieve this, the patches are not distributed uniformly. Instead, 10 patches are extracted from a random location inside the region of interest (ROI) of each image. The image's ROI is the part of the image where the entropy

is above a determined threshold or where retinal fluid is present. The patches are then randomly transformed by a rotation between 0 and 10 degrees, and horizontal flipping. Of the patches with no fluid, 75% of them were dropped. In Figure 3.2, it is possible to see the overlaying of the fluid masks and the ROI, with a red bounding box signaling a patch that would be used as input to train the U-Net.

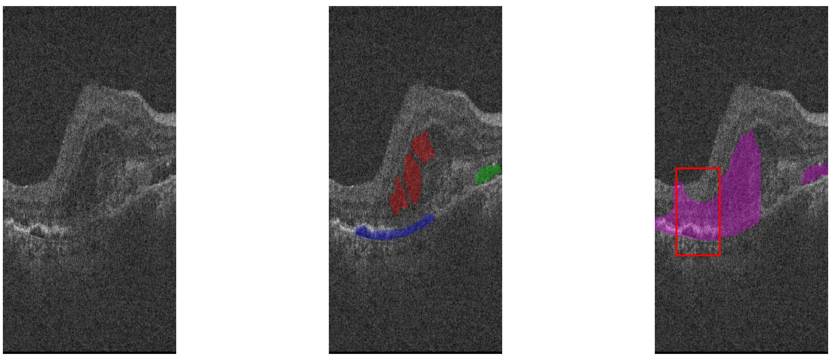

Figure 3.2: Cirrus B-scan (left), fluid masks overlay (middle) with IRF in red, SRF in green, and PED in blue, and the ROI mask overlaid in purple (right). The red bounding box signals a possible 256×128 patch that could be extracted.

In the Experiment 1.1, two sets of four training runs were performed, using each fold as validation. In both sets, the conditions were exactly the same, except the input patches, aiming to understand the effect that the random patch extraction has on the model's performances. The model was trained during 100 epochs with a batch size of 32, with no early stopping.

Experiment 1.2 In Experiment 1.2, the patch shape was changed from 256×128 to 496×512. By using such shape, the model receives a larger context of the B-scan as input, allowing it to learn the anatomic references that characterize and limit the fluids.

The patches used in this experiment were no longer randomly extracted from the ROI. Instead, the patches were extracted from top to bottom so that every section of the image would be present in at least one patch. In a Cirrus B-scan, with shape 1024×512, the first patch would correspond to section from $y = 528$ to $y = 1024$, while the second patch would be from $y = 32$ to $y = 528$. The last patch would then start on the bottom of the image, at $y = 0$, to $y = 496$. A representation of this process can be seen in Figure 3.3. The patches are extracted from top to bottom so that the retina and the fluid would not be split in two patches, damaging the quality of the input data.

In this experiment, due to the larger size of the patches that are being loaded, the batch size had to be reduced from 32 to 16. It was trained on 100 epochs with the same transformations as in the previous experiment and without early stopping.

Experiment 1.3 Another patch shape was experimented in Experiment 1.3. In this experiment, the images from different vendors were resized from their original dimensions to 496×512, the shape of the smaller images of the dataset, obtained with the Spectralis device.

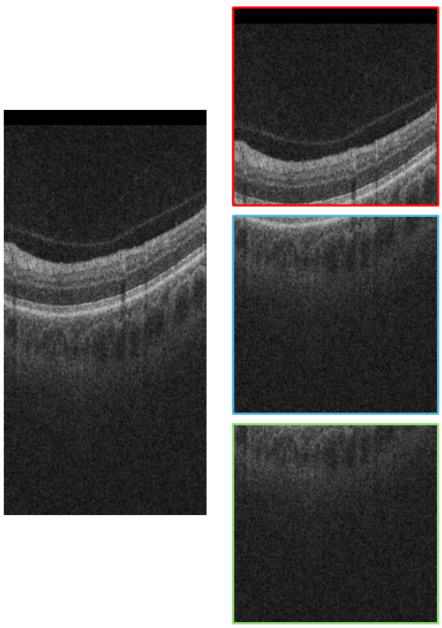

Figure 3.3: Cirrus B-scan and its respective three patches of shape 496 × 512.

The dimensions of the voxels in the OCT volumes change according to the device that was utilized to obtain the volume, resulting in images with different appearances across the vendors. For example, each voxel in the Cirrus volumes has an height of 1.95 μm, while each voxel in the Spectralis volumes has an height of 3.87 μm. For the same image, these differences in height lead to the same structures appearing bigger in Cirrus B-scans (see Figure 3.4). These differences across vendors makes the learning of the segmentation harder. Therefore, by resizing all the images to the same shape, the structures would have more consistent dimensions across vendors and the voxels roughly translated to the same dimensions, leading to a easier learning process for the model.

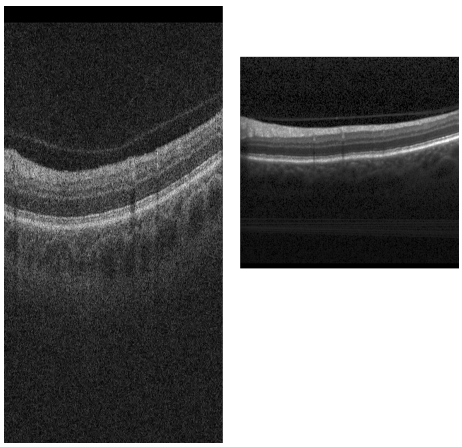

Figure 3.4: B-scan of the retinal layers in different patients, using Cirrus (left) and Spectralis (right) devices. In Cirrus, the retinal layers appear much larger than in Spectralis.

Then, vertical patches, of shape 496 $\times$ 128, were extracted from each B-scan. The number of patches extracted from each image was changed, experimenting with four (Figure 3.5), seven (Figure 3.6) and thirteen (Figure 3.7) patches. The advantage of extracting vertical patches is that each image contains both the complete retinal layer and the background. This does not happen in the previous experiments, where the patches are either too small to contain both background and the retinal layers (in Experiment 1.1) or the retinal layers are cropped during patch extraction (in Experiment 1.2, as seen in Figure 3.3).

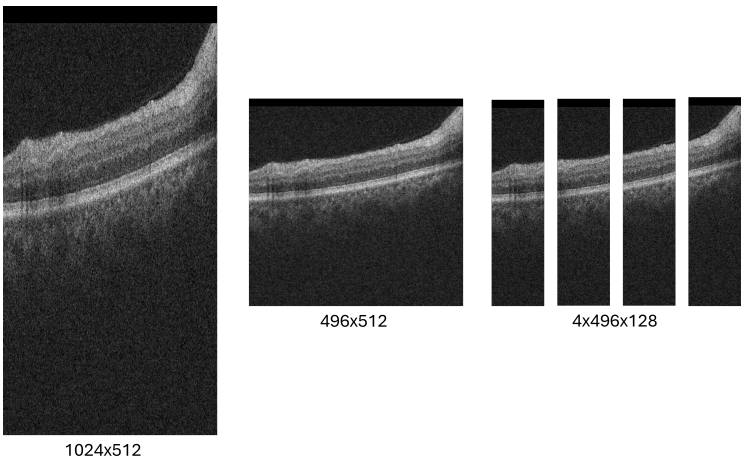

Figure 3.5: Four vertical patches of shape 496 $\times$ 128 extracted from a Cirrus B-scan.

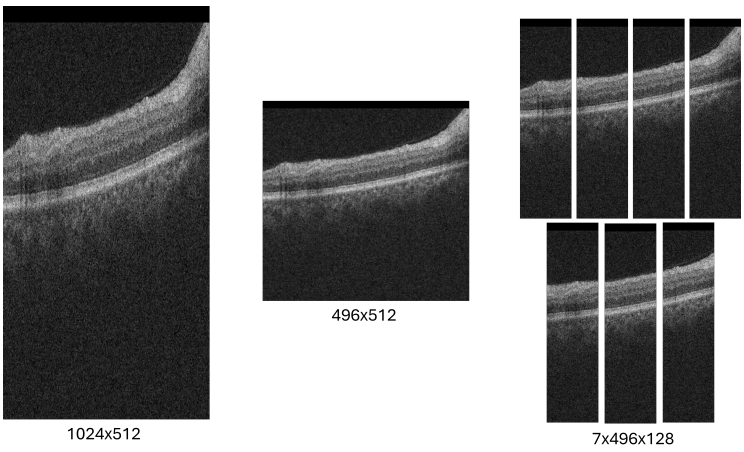

Figure 3.6: Seven vertical patches of shape 496 $\times$ 128 extracted from a Cirrus B-scan.

When using four vertical patches, the model was trained both for 100 and 200 epochs, maintaining a batch size of 32 and a maximum rotation of 10°, without early stopping.

Then, the model was trained using seven and thirteen patches for the best and worse performing folds when using four patches, while stopping early in case the validation loss did not progress 25 epochs after the minimum validation loss was encountered. This was used because the model progressed much faster when using seven and thirteen patches per B-scan, as it was trained in a

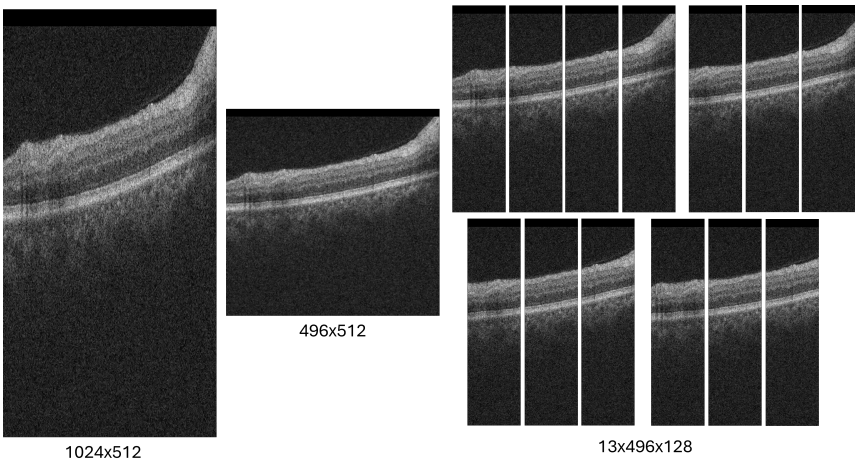

Figure 3.7: Thirteen vertical patches of shape 496 × 128 extracted from a Cirrus B-scan.

much larger number of images per epoch. Henceforth, it also required more computational power, which further motivated the early stopping.

Using seven vertical patches per image, which was the number of patches that performed best, three different rotations were experimented: no rotation, maximum rotation of 5°, and maximum rotation of 10°. These values were tested for a better understanding of how the rotation of the image affects the segmentation and the model's understanding of anatomic references. The model was trained on a minimum of 100 epochs, after which a patience of 25 epochs was applied. Therefore, if after 100 epochs, the model did not improve its validation loss for 25 consecutive epochs, then training would be interrupted.

Lastly, the model was trained using four patches and a maximum rotation of 5°, using the same early stopping criteria as in the last seven patches runs. This allowed for one last comparison between the two number of patches used in training, under the same conditions. The best model between those trained with four patches and those trained with seven patches was selected to infer on the reserved fold.

3.2.2.2 Experiment 2

The second experiment also involved multi-class segmentation of the retinal fluids. Contrasting with the first experiment, where the segmentation was done using a U-Net, three U-Nets were used in this experiment, one for each fluid. Each U-Net model focused only on the segmentation of one fluid, with one model for IRF, one for SRF, and one for PED.

One of the main problems with multi-class segmentation performed by binary models is the merging of the multiple masks. In this context, multiple fluid classes can be predicted for the same voxel, while only one of those can be correct. In this experiment, two alternatives were explored: order of priority, where the merging of the fluid masks follows a predefined hierarchy that determines which fluid takes precedence, and highest probability, where the assigned class is the one that was predicted with highest probability by its model.

Two different losses were used to regulate the model. Initially, the same loss as the one used in 3.2.2.1 Experiment 1, with only two classes (background and the fluid that would be segmented), and then, the weighted cross-entropy, using weights that balance the larger quantity of background voxels. This last loss is described in Equation 3.5, with $N = 2$.

When using the loss from 3.2.2.1 Experiment 1, each model was trained on two different splits: the split used in the multi-class segmentation experiments and a split created specifically for the segmentation of the fluid that was being segmented.

The balanced cross-entropy loss was tested on the best performing folds of the multi-class and IRF splits, for the segmentation of IRF, in the same conditions. However, since the results were much worse than those obtained with the initial loss, no more folds or fluids were considered.

All the models were trained with seven vertical patches extracted from each B-scan, on a minimum of 100 epochs, after which a patience of 25 epochs was applied, like what was done in the last runs of 3.2.2.1 Experiment 1. Similarly, the random transformations applied to the images consisted of horizontal flipping and a maximum rotation of 5°.

3.2.3 Intermediate Slice Synthesis

The objective of the subsequent experiments is to improve the resolution between slices, thus approximating the estimated fluid volume to the true value.

The intermediate slices were generated using the RETOUCH dataset as training and validation data. In this experiment, subvolumes that consist of overlapping triplets of consecutive slices, sampled with a step size of 1, were used, extracted as shown in Figure 3.8. The first and the last slice of these triplets were used for the generation of the middle slice. Consequently, it is possible to evaluate the generated slice in comparison to the original one, as done in other examples of the literature. For each volume, the number of potential subsets is then determined to be $n-2$, where n represents the number of slices within the same volume.

The generation of slices can be evaluated in specific metrics, as well as through qualitative assessment. To assess the efficacy of the generation model, the model utilized for fluid segmentation could be used for the estimation of the fluid's area in the generated image and to compare the resulting mask with the original image's mask. This comparison can be conducted using the Dice coefficient [14] (see Equation 3.2). However, this metric is insufficient for evaluating the generation performance, as it requires comparisons that encompass the entire slice and not just the fluid region. Examples of such metrics include the mean absolute error (MAE) [14, 42, 52], the peak signal-to-noise ratio (PSNR) [39, 41, 44, 46, 47, 49, 50, 51, 52], and the structural similarity index measure (SSIM) [39, 44, 46, 47, 49, 50, 51, 52].

The MAE and mean squared error (MSE) quantify the errors between the original image and the generated image. For every pixel, the difference between the value in the original image and the generated image is calculated. In MAE, the absolute value of this difference is calculated, and then the mean of all pixels in the image is computed. Meanwhile, in MSE, the difference is squared before computing the mean. In Equation 3.6 and Equation 3.7, MAE and MSE are described, respectively, where x_i is the intensity of the pixel of index i in the predicted image,

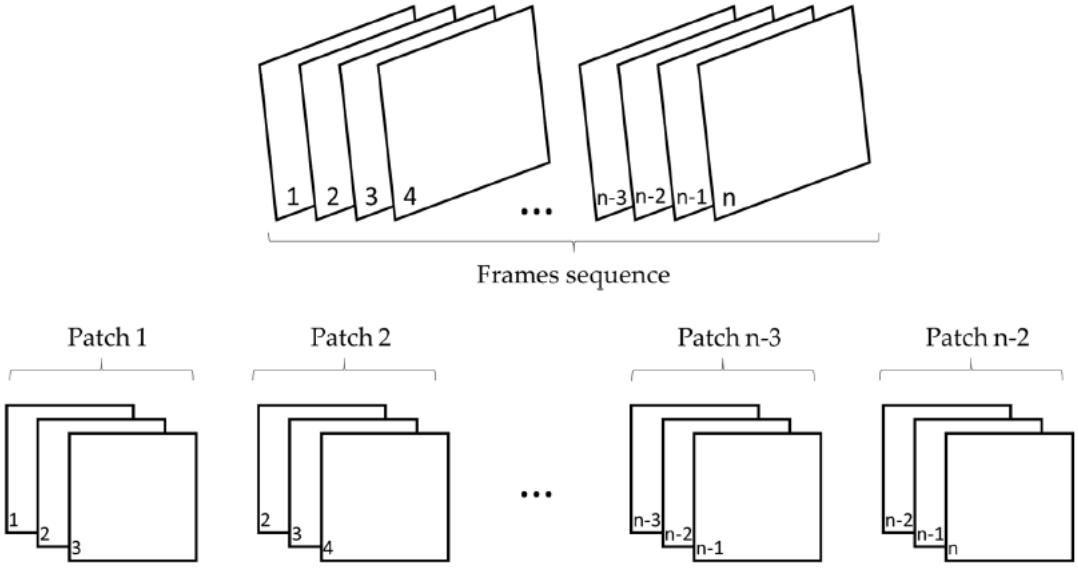

Figure 3.8: Scheme explaining the input data of the generative models. Each frame refers to B-scan from an OCT volume. Extracted from Tran and Yang [58].

while y_i is the intensity of the pixel with index i in the original image, and N is the number of pixels in the image [67, 68].

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|x_i - y_i| \tag{3.6}$$

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 \tag{3.7}$$

PSNR is a metric used to calculate the ratio between the maximum signal power (which corresponds to the maximum value of a pixel in the image) and the power of the distorting noise, which affects the quality of its representation. Therefore, the PSNR, described in Equation 3.8, is inversely proportional to the mean squared error. PSNR can also be understood as the representation of absolute error in dB [67]. It is important to note that in OCT the signal-to-noise ratio is low, due to the speckle present in the images. Therefore, a smaller PSNR is also expected, when compared with other imaging techniques that do not present as much speckle [2].

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{L^2}{\text{MSE}}\right) \tag{3.8}$$

While the previous metrics focus on the differences between two images at a pixel level, the SSIM is based on the perception of the image. This metric considers the change of perception in structural information, estimating the perceived quality of images and videos. SSIM measures the similarity between the original image and the generated. The SSIM is calculated as shown in Equation 3.9, where x and y represent the generated and the original images, respectively, so that

μ_x and μ_y are their local means, σ_x and σ_y are their standard deviations, while C_1 and C_2 are small constants that stabilize the division. The contrast sensitivity (CS) between the images x and y is represented by $CS(x,y)$ [67].

$$\text{SSIM}(x,y) = \left(\frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}\right) \cdot \left(\frac{2\sigma_{xy} + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}\right) = \left(\frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}\right) \cdot CS(x,y) \tag{3.9}$$

Since the best performing models in segmentation resized the images to 496 $\times$ 512, the images were generated to match those dimensions. Therefore, regardless of the device utilized to obtain the OCT volume, all its B-scans were resized to 496 $\times$ 512 for both experiments of image generation.

3.2.3.1 Experiment 3

In the first experiment focused on intermediate slice synthesis, a GAN was used. The underlying principle of a GAN, originally proposed by Goodfellow et al. [69], is based on a competitive game between two networks. The generator network starts with the first and last slice of a subvolume, which is composed of three consecutive B-scans from an OCT scan, and aims to generate the intermediate slice. In contrast, the discriminator network is trained to distinguish between the generated and real slices. When the discriminator correctly labels generated slices as fake, the generator is penalized, motivating it to fool the discriminator and consequently improving its generation, resulting in outputs more similar to the real inputs. However, the discriminator network loss also penalizes misclassifications, dependent on the probability of the prediction. As a result, as the generator improves, so does the discriminator [70]. The overall framework for GANs is illustrated in Figure 3.9.

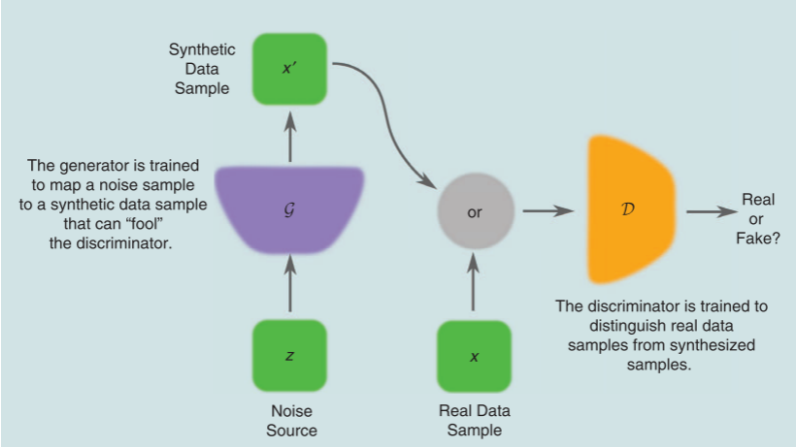

Figure 3.9: Example of a GAN framework, where $\mathscr{D}$ is the discriminator and $\mathscr{G}$ is the generator [71].

The GAN implemented by Tran and Yang [58] was used to generate the intermediate slices of the OCT volumes. This framework is used in the interpolation of intermediate slices in video and is trained in patches of 64 × 64.

In the original implementation, one patch is randomly extracted from the triplet of images and used in training. Due to the much smaller quantity of data available in OCT, all the possible disjoint 64 × 64 patches were extracted from each B-scan. The extraction was done from top to bottom and in the last row of slices, the image was padded until it had 64 pixels. An example of the patches extracted from a Cirrus B-scan can be seen in Figure 3.10. By methodically extracting the patches, triplets are easily created by accessing patches of the same index in the three consecutive images.

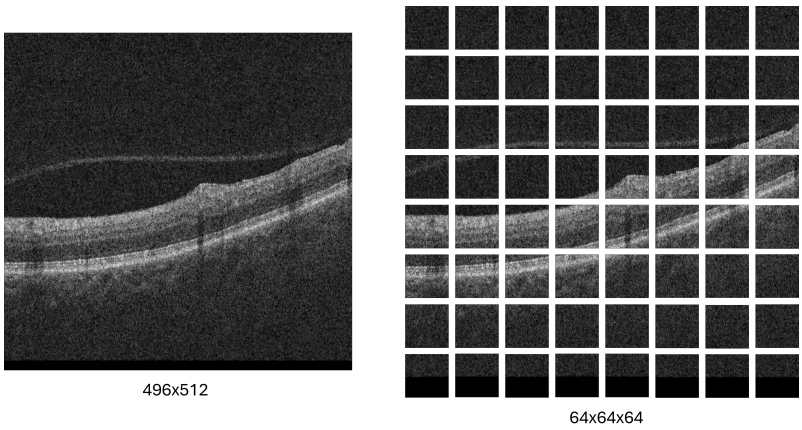

Figure 3.10: Patches with shape 64 × 64 extracted from a Cirrus B-scan which was resized to 496 × 512.

The generator of the GAN is composed of contracting and expanding path. In the contracting path, convolutions are applied to the images and feature maps, followed by batch normalization, and a leaky ReLU activation function. After the images are downsampled to 512 × 8 × 8, reaching the bottleneck, the expanding path begins, where deconvolutions are applied, followed by batch normalization, and a leaky ReLU. Finally, after the last deconvolution, the hyperbolic tangent function is used as the final activation function, resulting in an output of size 64 × 64, in range -1 to 1. An illustrative scheme of the generator can be seen in Figure 3.11.

To match the needs of our application, some changes had to be made in the generator regarding the input shape. As shown in Figure 3.11, the input has six channels, one red, one green, and one blue (RGB) for each input patch. Similarly, the output has three channels, for the generated middle patch. In our application, each patch in the input only had one channel, since the OCT B-scans are images in gray scale, instead of RGB, as in the original implementation. Therefore, the output only had one channel.

The final activation function, the hyperbolic tangent, outputs values between -1 and 1. Since the output images were compared to images in range 0 to 1, the final activation function was changed to the sigmoid. This function converts the values output from the last convolution to the range of 0 to 1, where it can be compared to the GT images.

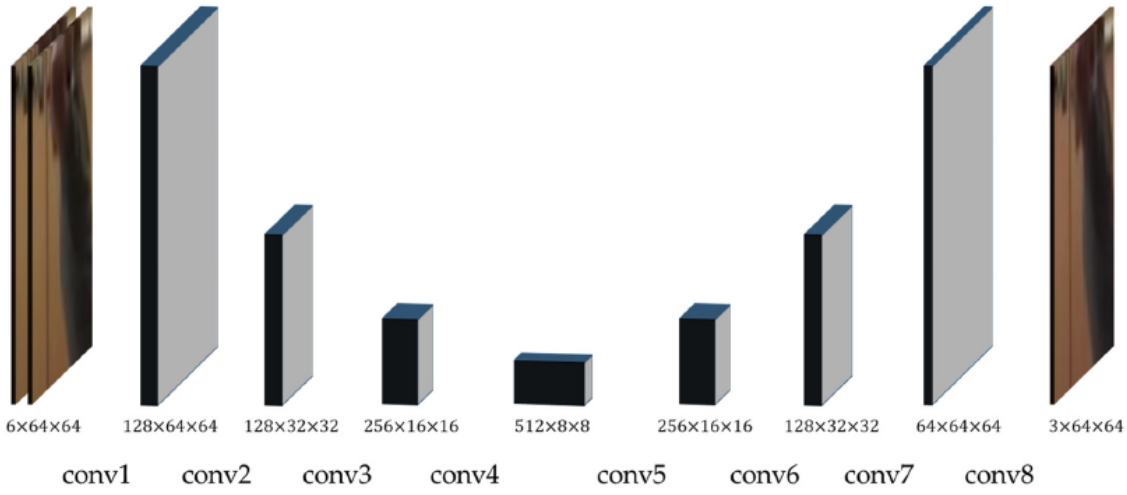

Figure 3.11: Architecture of the generator used in the GAN. It has a contracting and an expanding path, making it a U-Net like network [58].

The GAN's discriminator receives as input a patch of shape 64×64, and outputs a probability of the input patch being real. The discriminator is composed of five consecutive convolutions. The first convolution is followed by a leaky ReLU activation function. The second, third, and fourth convolutions are followed by a batch normalization and a leaky ReLU activation function. After the last convolution, the sigmoid is applied and converts the final output to a value between 0 and 1 that represents the probability of the image being real. In Table 3.4, the layers that compose the discriminator and the generator are explained, including the shape of the outputs.

In the training of a GAN, both the generator and the discriminator are being trained sequentially and independently. First, the generator, which receives the previous and following patch of the input triplet, attempts to generate the intermediate patch. The generated image is then compared to the original image, using the generator loss, which is then used in the updating of the generator weights. The generator loss is composed of four components: the adversarial, the MAE, the multi-scale SSIM (MS-SSIM), and the gradient difference loss (GDL). The overall loss function is described in Equation 3.10, where $\lambda_{\text{adv}} = 0.05$, $\lambda_{\text{MAE}} = 1.0$, $\lambda_{\text{MS-SSIM}} = 6.0$, and $\lambda_{\text{GDL}} = 1.0$, representing the weights of each loss component.

$$\mathscr{L}_{\text{Gen}} = \lambda_{\text{adv}} \times \mathscr{L}_{\text{adv}} + \lambda_{\text{MAE}} \times \mathscr{L}_{\text{MAE}} + \lambda_{\text{MS-SSIM}} \times \mathscr{L}_{\text{MS-SSIM}} + \lambda_{\text{GDL}} \times \mathscr{L}_{\text{GDL}} \tag{3.10}$$

The adversarial loss is used in the evaluation of how good the output from the generator fools the discriminator. This evaluation is done using the binary cross-entropy (BCE). To calculate this, the generated image is input into the discriminator, which then outputs the probability of being real. Afterwards, the BCE is calculated for the predicted probability and 1, the label of a real image. The better the generator fools the discriminator, the closer the output of the discriminator is to one, and, therefore, the closer the adversarial loss is to 0. The adversarial loss is explained in Equation 3.11, where $\mathscr{D}$ is the discriminator, x is the generated image, and y is the label that

Table 3.4: Layers that compose the generator and the discriminator. Each convolution is represented by Conv2d(K, OC, S), where K is the kernel size, OC is the number of output channels, and S is the stride. The same notation is used in deconvolutions, represented by TransposedConv2d. The output size is shown following C × H × W notation, where C is the number of channels, H is the height, and W is the width. The inputs have shape $1 \times 64 \times 64$. Adapted from Tran and Yang [58].

Generator		
Layers	**Details**	**Output Size (C × H × W)**
1	Conv2d(3, 128, 1), BatchNorm2d, LeakyReLU	128 × 64 × 64
2	Conv2d(4, 128, 2), BatchNorm2d, LeakyReLU	128 × 32 × 32
3	Conv2d(4, 256, 2), BatchNorm2d, LeakyReLU	256 × 16 × 16
4	Conv2d(4, 512, 2), BatchNorm2d, LeakyReLU	512 × 8 × 8
5	TransposedConv2d(4, 256, 2), BatchNorm2d, LeakyReLU	256 × 16 × 16
6	TransposedConv2d(4, 128, 2), BatchNorm2d, LeakyReLU	128 × 32 × 32
7	TransposedConv2d(4, 64, 2), BatchNorm2d, LeakyReLU	64 × 64 × 64
8	TransposedConv2d(1, 1, 1), Sigmoid	1 × 64 × 64
Discriminator		
Layers	**Details**	**Output Size (C × H × W)**
1	Conv2d(4, 64, 2), LeakyReLU	64 × 32 × 32
2	Conv2d(4, 128, 2), BatchNorm2d, LeakyReLU	128 × 16 × 16
3	Conv2d(4, 256, 2), BatchNorm2d, LeakyReLU	256 × 8 × 8
4	Conv2d(4, 512, 2), BatchNorm2d, LeakyReLU	512 × 4 × 4
5	Conv2d(4, 1, 1), Sigmoid	1 × 1 × 1

indicates that the image is real. It is important to note that in the generator training, the images that are input to the discriminator are detached, not contributing to the updating of the discriminator weights in this step.

$$\mathscr{L}_{\text{adv}}(\mathscr{D}(x), y) = \mathscr{L}_{\text{BCE}}(\mathscr{D}(x), y) = -\left[y\log(\mathscr{D}(x)) + (1-y)\log(1-\mathscr{D}(x))\right] \tag{3.11}$$

The MAE loss, also referred to as L_1 loss, performs a pixel-by-pixel comparison between the generated image, x, and the real image, y, as described in Equation 3.6, where i is the index of a pixel. While this loss gives an insight of how similar the images are, on average, this loss can be deceiving, since the model can blur the output to attain better MAE values. For this reason, this loss must be combined with other reconstructive losses, such as the MS-SSIM and the GDL.

The MS-SSIM loss seeks to preserve the structural similarity, at different scales, between the real and the generated image, facilitating a smoother output. This loss, originally suggested by Wang et al. [72], is described in Equation 3.12, while the MS-SSIM used in this implementation is explained in Equation 3.13, using the concepts of SSIM and CS explained in Equation 3.9. This

version of the MS-SSIM is faster than the original implementation [72], while using the same array of weights β and number of levels M, which is set to 5. Therefore, the MS-SSIM corresponds to the product of the contrast sensitivity in the image for first four levels and the SSIM of the image in the last level, all raised to the power of the respective level's weights.

$$\mathscr{L}_{\text{MS-SSIM}}(x,y) = 1 - \text{MS-SSIM}(x,y) \tag{3.12}$$

$$\text{MS-SSIM}(x,y) = \prod_{j=1}^{M-1} [\text{CS}_j(x,y)]^{\beta_j} \cdot [\text{SSIM}_M(x,y)]^{\beta_M} \tag{3.13}$$

The last component of the loss is the GDL, proposed originally by Mathieu et al. [73]. This component is used to reduce the motion blur in the generated images, a problem in video datasets. In this loss, the relative difference of neighboring pixels between the generated and true images is considered, as shown in Equation 3.14. In this equation, i and j are the index of row and column, respectively, that identify a pixel of the image, with α set to 2.

$$\mathscr{L}_{\text{GDL}}(x,y) = \sum_{i,j} \left(\left| |x_{i,j} - x_{i-1,j}| - |y_{i,j} - y_{i-1,j}| \right|^{\alpha} + \left| |x_{i,j} - x_{i,j-1}| - |y_{i,j} - y_{i,j-1}| \right|^{\alpha} \right) \tag{3.14}$$

After the images are generated and evaluated using the previously defined generator loss, two images are input, subsequently, to the discriminator, with one of them being fake while the other is real. The discriminator outputs the probability of each image being real and its result is compared to the true label of each image using the BCE. The BCE is calculated for the probabilities predicted by the discriminator and the image's respective label as described in Equation 3.11. This is done for the fake image and for the real image. The mean between the BCE calculated for the fake image and the BCE computed for the real image is the discriminator loss.

The GAN was trained in 250 epochs, using a batch size of 32 and 2×10^{-4} as the learning rate. The selected optimizer was Adam, with $\beta_1 = 0.5$ and $\beta_2 = 0.999$.

3.2.3.2 Experiment 4

As in the previous experiment, the intermediate slice was generated using the first and last slices of a subvolume that consists of three consecutive B-scans from an OCT scan. However, in this experiment, inspired by the work of Nishimoto et al. [43], the intermediate slice was generated using a U-Net. While the U-Net is more commonly applied in segmentation, as seen in the reviewed literature, Nishimoto et al. [43] apply it to generate the intermediate slices of a subvolume. The U-Net receives the edge slices as input and forces the output of the intermediate ones. In the paper [43], this was tested for three, four, and five slices. However, in this experiment it was utilized to generate a single intermediate slice.

In this experiment, the whole image is input to the network and for that reason the batch size was set to 8. The optimizer utilized was Adam with a learning rate of 2×10^{-4} and the model was trained for 200 epochs.

3.2.4 Fluid Volume Estimation

The estimation of fluid volume was done using the optimal segmentation and intermediate slice generation models. The GAN was used in the generation of the intermediate slices in the OCT volumes, while the best performing multi-class segmentation U-Net model inferred the segmentation masks to both the unaltered volumes and the volumes with generated slices.

The OCT scans used in the fluid volume estimation experiments were the ones from the RETOUCH dataset that composed the reserved fold in the segmentation and generation experiments and the ones from the private dataset obtained in Hospital São João. These volumes were selected because no model was trained or validated on them, allowing an insight of both models generalization on unseen data and how the increase in resolution affects the total fluid volume.

The area of each fluid in each OCT scan was estimated considering the resolution of each OCT scan, which varies according to the device utilized to obtain the OCT volume. Afterwards, the area is multiplied by the axial distance (half the axial distance to the previous slice plus half the axial distance to the following slice) to obtain the volume of fluid per slice. In the first and last slice of an OCT volume, the area is multiplied by half of the axial distance (half the axial distance to the neighboring slice). The total volume of fluid in an OCT scan can be estimated by summing the fluid volumes of all individual B-scans. This allows the volume estimation of IRF, SRF, and PED, as well as the overall fluid volume in the OCT scan.

The total volume of fluid from class c in a slice of index s is defined in Equation 3.15. The slice belongs to an OCT volume obtained using device D and its total number of B-scans is defined as S. In this equation, H_D and W_D are the height and width of a voxel, respectively, obtained with device D, while $d_{D,s,s+1}$ is the axial distance between the slice of index s and the slice of index $s+1$, a value that depends on the device D characteristics. The variable l_i is the label attributed to the voxel of index i. Like the variable c, l can be one of the following classes: $\{0,1,2,3\}$, which respectively correspond to background, IRF, SRF, and PED. Meanwhile, the total volume of fluid from a class c in an OCT scan obtained with device D is described by Equation 3.16, and consists of the sum of the fluid's volume obtained in each B-scan that compose the OCT.

$$
\begin{aligned}
f_{c,s,D} &= \sum_i (v_{i,s,D} \times y_{i,c}) \quad \text{where:} \\
v_{i,s,D} &= \begin{cases} 0.5 \times H_D \times W_D \times d_{D,s,s+1} & \text{if } s = 0 \\ 0.5 \times H_D \times W_D \times d_{D,s,s-1} & \text{if } s = S \\ 0.5 \times H_D \times W_D \times d_{D,s,s-1} + 0.5 \times H_D \times W_D \times d_{D,s,s+1} & \text{otherwise} \end{cases} \\
y_{i,c} &= \begin{cases} 1 & \text{if } l_i = c \\ 0 & \text{if } l_i \neq c \end{cases}
\end{aligned} \tag{3.15}
$$

$$
F_{c,D} = \sum_s^S f_{c,s,D} \tag{3.16}
$$

The fluid volumes resulting from both experiments were compared. Since there is no true value for the fluid quantity in the OCT scans, the results were compared with each other. Therefore, the results would be deemed satisfying in case they do not vary more than an order of magnitude between each other. In case a significant difference was observed, the generated images and their respective masks were analyzed, in order to understand what is causing the observed difference between experiments.

3.2.4.1 Experiment 5

In this experiment, the fluid volumes were calculated for the OCT scans without the generated slices. The best segmentation model was utilized to segment the fluid in three classes and the volume was estimated for each class as described. The results from this experiment allow the comparison with the values obtained in the following experiment, where slice generation was used.

3.2.4.2 Experiment 6

This experiment consisted of the fluid volume estimation in OCT scans with generated images. The model used in segmentation was the same as in the previous experiment, which predicted the fluid masks for all the slices. From the predicted fluid masks, the fluid volume was estimated and compared with those obtained in the previous experiment.

Chapter 4

Results and Discussion

This chapter presents the results from the experiments described in the 3 Methods chapter. These outcomes are organized in the following sections as done in the Methods chapter: "4.1 Fluid Segmentation", "4.2 Intermediate Slice Synthesis", and "4.3 Fluid Volume Estimation". After showing the results from the experiments, the factors influencing them are discussed, while providing visual examples of the models' performances. The results are also compared with other similar approaches in literature, approaching the different methods that lead to different results.

4.1 Fluid Segmentation

In this section, the results from the experiments performed in multi-class fluid segmentation are shown. This includes all the runs made in 3.2.2.1 Experiment 1 and 3.2.2.2 Experiment 2. In these sections, the resulting Dice coefficients are displayed in tables. Each value corresponds to the mean Dice coefficient computed across all slices present in the validation OCT volumes. The results are shown for each fluid (IRF, SRF, and PED) both grouped by vendor and across all vendors. There is also a column that contains the Dice coefficient when considering all the fluids as a single binary label. Unless specified otherwise, the Dice coefficient calculated for a fluid considers all the slices and not just those that contain that fluid. The values highlighted in bold are the best values obtained in each column.

These values are presented for each run, which follows the conditions described in the 4.1 Fluid Segmentation section. Every four runs that are completed using a different validation fold but follow the same conditions are grouped in a single row, by calculating their mean (row "Set"). From the 5-fold split, fold 1 was reserved, while the remaining four (0, 2, 3, and 4) are used in training and validation. The fold selected for validation in each run appears in the table's validation fold ("VF") column. For example, if the fold in the "VF" column is 2, then the folds used in training were 0, 3, and 4.

In the following subsections, images displaying the segmentation performed by the models and the respective GT are shown. In these masks, IRF is represented in red, SRF in green, and PED in blue.

4.1.1 Experiment 1

4.1.1.1 Experiment 1.1

The results from the first experiment performed, Experiment 1.1, are shown in Table 4.1. In this experiment, for each validation fold, two runs were made, while keeping the same conditions. The only difference is the input data that, as it is extracted randomly as described in 3 Methods, is different for every run.

Table 4.1: Dice scores for every vendor and fluid for the runs done in Experiment 1.1. The conditions were the same for both sets except the extracted patches that are different in every run due to the random process of extracting them.

Runs	**VF**	**Cirrus**			**Spectralis**			**Topcon**			**IRF**	**SRF**	**PED**	**Fluid**
		IRF	**SRF**	**PED**	**IRF**	**SRF**	**PED**	**IRF**	**SRF**	**PED**				
Run 1	2	0.138	0.089	0.072	0.255	0.331	0.163	0.259	0.446	0.056	0.200	0.254	0.083	0.163
Run 2	3	0.138	0.290	0.236	0.264	0.670	0.652	0.389	0.532	0.284	0.252	0.445	0.327	0.303
Run 3	4	0.209	0.158	0.024	0.255	0.451	0.310	0.286	0.595	0.151	0.249	0.386	0.117	0.173
Run 4	0	0.166	0.179	0.041	0.307	0.343	0.243	0.371	0.376	0.052	0.266	0.280	0.081	0.165
Set 1	-	0.16	0.18	0.09	0.27	0.45	0.34	0.33	0.49	0.14	0.24	0.34	0.15	0.20
Run 5	2	0.106	0.152	0.085	0.290	0.406	0.256	0.433	0.443	0.065	0.250	0.296	0.110	0.175
Run 6	3	0.370	**0.297**	0.454	**0.410**	**0.735**	**0.673**	0.205	**0.821**	**0.672**	0.317	**0.566**	0.572	0.299
Run 7	4	**0.590**	0.265	**0.666**	0.296	0.372	0.498	**0.687**	0.683	0.555	**0.593**	0.460	**0.596**	**0.420**
Run 8	0	0.321	0.259	0.067	0.352	0.514	0.467	0.410	0.610	0.152	0.249	0.386	0.117	0.173
Set 2	-	**0.35**	**0.24**	**0.32**	**0.34**	**0.51**	**0.47**	**0.43**	**0.64**	**0.36**	**0.38**	**0.44**	**0.36**	**0.27**

By looking at the table, a few conclusions can be drawn. First, the difference in performance between runs using the same training OCT volumes is evident. Despite some values being similar for the same training volumes, this trend becomes evident when looking at the mean values in the rows "Set", where a significant difference is noted, especially in the Cirrus vendor and the PED fluid. It is also evident that some VFs are more consistent than others. For example, for different extracted patches, the models evaluated on validation fold 2 and validation fold 0 presented similar results, while those evaluated in validation fold 3 showed significant differences when the input patches were changed.

The second conclusion is that, overall, the models are not performing well, as the Dice results are low for every VF. These values are specially low in Cirrus, but better both in Spectralis, and Topcon. One of the reasons the model performed so poorly is due to the input it was receiving. Most of the extracted patches do not capture the transitions from background to retina and from retina to the choroid due to its small size. These transitions are of great importance in fluid segmentation in OCT scans since these transitions represent, among other concepts, the boundaries of the region where fluid can appear. In case the model does not understand these anatomic boundaries, segmentation can be performed outside the retina, which worsens the Dice coefficient.

Despite the performances not being good in every vendor, it is worse in Cirrus. In the literature, it is common to see worse performances in the images obtained with Cirrus and Topcon due to the larger presence of speckle noise in them. However, the worse performance in Cirrus is this experiment was due to the patches extracted from the volumes obtained with devices. The B-scans in these volumes present a larger vertical resolution than those obtained with Topcon and Spectralis

devices. Therefore, when extracting a patch of the same size across all devices, each patch from a Cirrus scan captures a smaller area of the retina. This translates to an harder understanding of the transition between background and retina and between retina and choroid, as shown in Figure 4.1.

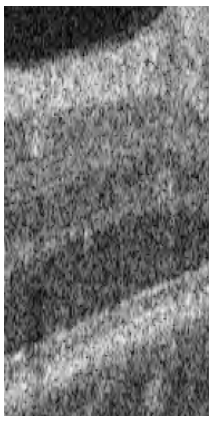

Figure 4.1: Example of a patch extracted from a Cirrus OCT volume used in Experiment 1.1. In this patch, while the background is noticeable due to its darker shade, the choroid is harder to be identified by an observer (or a model) due to the lack of context.

In Figure 4.2 it is shown a segmentation performed by the model trained in Run 1 (see Table 4.1). In this figure it becomes evident that the model learned which areas can be segmented inside the retina, but does not understand how the handle the regions further away. For example, in the area significantly above the retina the background is labeled as SRF. However, the background region closer to the retina is not so frequently labeled as any fluid, since it appears in the patches input to the model. The same problem occurs with the oversegmentation of PED in the choroid region, which does not appear in the input patches significantly.

Oversegmentation beyond the retinal bounds is not exclusive to the Cirrus volumes, as it also appears in the OCT scans from other vendors. This suggests that the issue is primarily due to the small patch size rather than the smaller retinal area captured in Cirrus patches, despite this amplifying the problem.

In the same figure, it is seen that the model has not learned the anatomical relationships between fluids, since it segmented IRF and PED close to each other. This occurs because the model has not learned the relationship between the retinal landmarks and the fluid types due to the small sized inputs.

4.1.1.2 Experiment 1.2

The resulting Dice coefficient values obtained in Experiment 1.2, where patches of shape 496 $\times$ 512 were used, are shown in Table 4.2.

The information resumed in this table allows the comparison between the performance in models that used smaller patches in Experiment 1.1 with those that used larger patches in this Experiment.

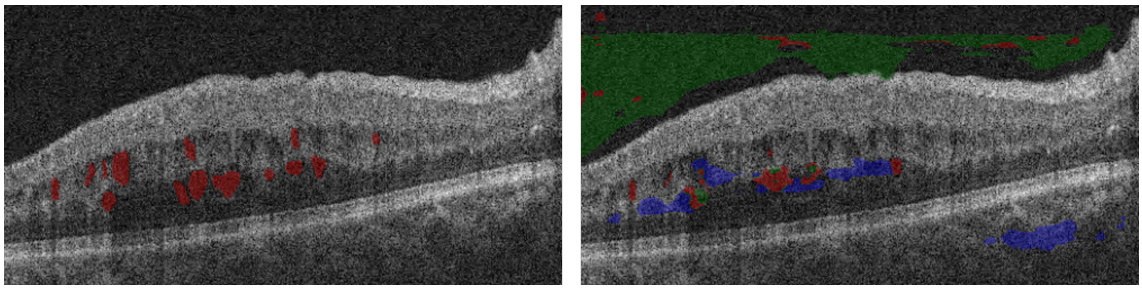

Figure 4.2: Example of a poor segmentation made by the model trained in Run 1 (right). In the left, the GT mask for the same image is shown.

Table 4.2: Dice scores for every vendor and fluid for the runs done in Experiment 1.2.

Runs	**VF**	**Cirrus**			**Spectralis**			**Topcon**			**IRF**	**SRF**	**PED**	**Fluid**
		IRF	**SRF**	**PED**	**IRF**	**SRF**	**PED**	**IRF**	**SRF**	**PED**				
Run 9	2	0.291	0.450	0.281	0.472	0.638	0.394	0.505	0.647	0.573	0.396	0.551	0.400	0.393
Run 10	3	**0.586**	**0.619**	**0.727**	0.482	**0.780**	**0.698**	**0.749**	**0.793**	**0.788**	**0.627**	**0.711**	**0.744**	**0.667**
Run 11	4	0.281	0.453	0.415	0.429	0.503	0.322	0.228	0.532	0.324	0.278	0.494	0.363	0.296
Run 12	0	0.242	0.334	0.336	**0.551**	0.564	0.423	0.321	0.643	0.407	0.325	0.488	0.377	0.339
Set 3	-	0.35	0.46	0.44	0.48	0.62	0.46	0.45	0.65	0.52	0.41	0.56	0.47	0.42

By comparing the "Set" rows, it becomes evident an overall increase in performance when using larger patches. In fact, it is observed an increase in almost all the mean values when compared to the best performing set in Experiment 1.1. This increase is larger than one decimal point in some columns, and the largest improvements occur in the volumes obtained using the Cirrus device and when segmenting SRF or PED.

The improvement in performance in Cirrus devices can be explained with the more contextual information given. In Experiment 1.1, the Cirrus patches input to the model were not big enough for the network to capture and understand the retinal borders or the relationship between them and the fluids, thus leading to oversegmentation beyond the retina, as explained. In Experiment 1.2, as the same patch covers the retinal layer and the background simultaneously, the model has learned to not segment beyond the retina. In Figure 4.3, the B-scan shown in Figure 4.2, is segmented by the model trained in Run 9 and it is possible to see that the labeling of the background as fluid is no longer happening.

This also partially justifies the significant improvement in the SRF and PED Dice coefficient. By providing inputs with enough context, the model is no longer segmenting these fluids outside the retina. However, in Figure 4.3 it is also seen that the model is no longer confusing the fluids in the retina, as it no longer segments the PED close to the IRF. This further justifies that the model is learning the anatomical references associated with PED, as confirmed in Figure 4.4.

Lastly, it is important to note that the IRF did not improve as much as the other fluids. This happens because the input patches in Experiment 1.1 where extracted mainly from the retina, providing all the anatomical context needed for the IRF segmentation.

In Figure 4.4, it is seen that the model confuses the choroid with the retina, as it labels parts of that region as IRF and PED. The likely cause for this confusion is that the input patches are

often cropped in the middle of the retinal layer (as illustrated in Figure 3.3), which leads to the model not understanding the position of the choroid relative to the retina. This inaccurate IRF and PED segmentation is probably based on the visual resemblance between the structures seen in the choroid and the fluid pockets in the retina. This indicates that the model does not know the location of the choroid.

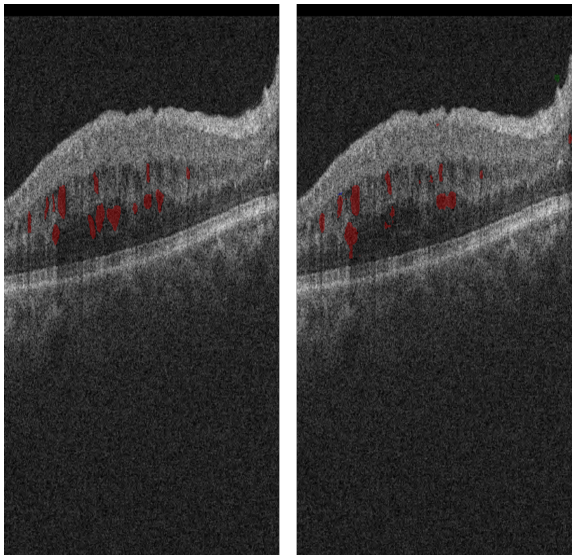

Figure 4.3: Example of the segmentation done by the model trained in Run 9 (right) and its respective GT mask (left). The B-scan segmented is the same as in Figure 4.2.

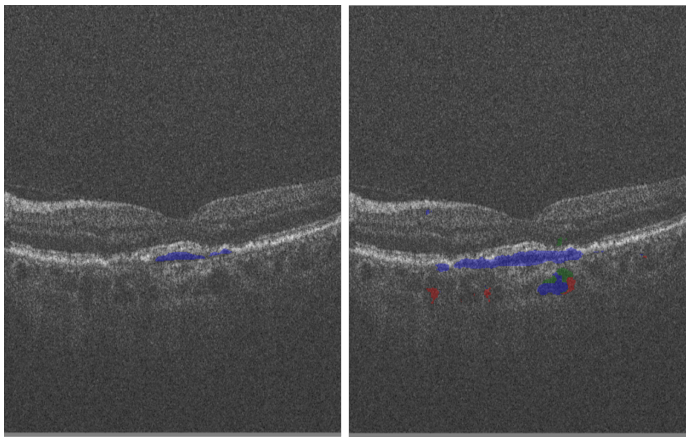

Figure 4.4: Example of the segmentation done by the model trained in Run 12 (right) and its respective GT mask (left). It is noticeable that the model confuses the choroid with the retina, as segmentation of IRF and PED is performed in the choroid.

4.1.1.3 Experiment 1.3

Experiment 1.3 contains all the runs that were performed using vertical patches extracted from resized B-scans as input of the multi-class segmentation U-Net. In the first two sets, shown in

Table 4.3, the models were trained using four vertical patches, obtained as explained in the 3 Methods chapter. In this table, the "Set 4" is composed of runs that were trained on 100 epochs, while the runs in "Set 5" were trained on up to 200 epochs with a 100 epoch patience.

Table 4.3: Dice scores for every vendor and fluid for the runs done using four vertical patches extracted from each B-scan. In "Set 4", the models were trained in 100 epochs, while in "Set 5" the models were trained on up to 200 epochs with a 100 epoch patience. The transformations applied in these sets were the same: horizontal flipping and a maximum rotation of 10°.

Runs	**VF**	**Cirrus**			**Spectralis**			**Topcon**			**IRF**	**SRF**	**PED**	**Fluid**
		IRF	**SRF**	**PED**	**IRF**	**SRF**	**PED**	**IRF**	**SRF**	**PED**				
Run 13	2	0.411	0.665	0.498	0.654	0.735	0.611	0.731	0.743	0.519	0.563	0.704	0.526	0.581
Run 14	3	0.390	0.520	0.217	0.403	0.564	0.429	0.378	0.754	0.477	0.388	0.614	0.350	0.386
Run 15	4	0.792	**0.846**	**0.883**	0.732	**0.901**	0.733	0.615	0.758	0.656	0.707	0.815	0.765	0.652
Run 16	0	**0.810**	0.834	0.715	0.779	0.857	0.753	0.783	**0.924**	**0.882**	**0.795**	**0.871**	**0.783**	**0.709**
Set 4	-	**0.60**	**0.72**	0.58	0.64	**0.76**	0.63	0.63	**0.79**	0.63	0.61	**0.75**	0.61	0.58
Run 17	2	0.522	0.784	0.703	**0.805**	0.765	**0.865**	**0.828**	0.879	0.835	0.677	0.813	0.777	0.684
Run 18	3	0.509	0.654	0.598	0.590	0.806	0.755	0.777	0.787	0.792	0.621	0.729	0.697	0.622
Run 19	4	0.377	0.387	0.398	0.478	0.624	0.398	0.483	0.729	0.448	0.436	0.567	0.420	0.399
Run 20	0	0.753	0.697	0.696	0.779	0.844	0.783	0.691	0.720	0.671	0.735	0.731	0.702	0.637
Set 5	-	0.54	0.63	**0.60**	**0.66**	0.76	**0.70**	**0.69**	0.78	**0.69**	**0.62**	0.71	**0.65**	**0.59**

The results of the sets shown in Table 4.3 can be compared within each other and to the results obtained in the sets of previous experiments.

When comparing the results between sets, it is noticeable that the runs with the best scores are mostly located in the "Set" that trained less epochs. For, the model validated on fold 4 (Run 15) sees a significant decrease in performance when trained for 100 more epochs (Run 19). When looking at the models validation loss per epoch, the lowest value is attained commonly before the 100 epochs. Since the model that is saved in each run is the model that attains the lowest loss on validation data, this means that the models trained in 100 or 200 epochs should perform equally, as long as the lowest validation loss occurred prior to the 100 epoch mark.

However, looking at Figure 4.5, where the training and validation loss curves for the Run 13 and Run 17 are shown, similar trends and behavior is seen in both models. This does not reflect to similar performances in the same validation data, as shown in Table 4.3. In fact, a substantial increment is seen in Run 17.

To justify this behavior, it is important to remind that the training of a neural network has some randomness associated with it. One aspect that affects the model performance despite using the same data and architecture is the weight initialization. The random initialization of weights affects the trajectory of gradient descent, resulting in different convergences for different runs. Another factor that affects the model learning is the data that composes each batch. The images that form a batch are randomly fetch from the available input images. This create batches with different distributions of characteristics, affecting the model's view of the data. Data augmentation, which randomly selects images to apply the transformations as they are fetch, also influences the model performance. Lastly, there are other factors that have a much smaller impact such as the optimizer behavior and non-deterministic operations at the GPU level [74, 75].

In Appendix A, Table A.1 shows the results for multiple models that were trained on the same conditions, aiming to give an insight on how the randomness associated with the U-Net affects the models performances. Five models were validated on fold 2, while nine models were validated in fold 3. The results obtained corroborate the differences seen in Table 4.3, where models trained on the same conditions for more epochs can obtain significantly worse Dice scores in validation.

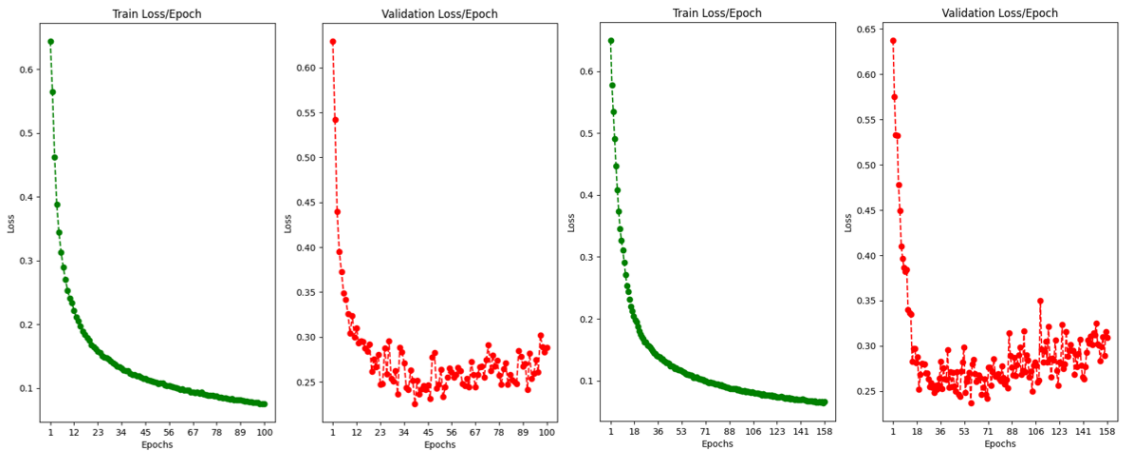

Figure 4.5: Training and validation losses in Run 13 (left) and Run 17 (right). Despite reaching the validation loss minimum in a similar number of epochs and having comparable training and validation loss curves, the performance is really different in Table 4.3.

Despite the differences in the results shown in Table 4.3, both sets attained better performances than the best results in the previous experiments. The improvements were significant, registering increases above 0.2 in some metrics when compared with the results in Experiment 1.3. Every vendor and every fluid seen a considerable increase of performance.

This overall increment can be attributed to two changes: training on vertical patches and resizing the images. By training the models on vertical patches, the model can focus on the finer details within each region. This is important because the retinal fluid segmentation relies heavily on the retina's characteristics, a region that occupies a small portion of the image. Despite the input being smaller than in Experiment 1.3, the patches could still capture the transitions between the retina and the choroid or background. Since the input size is smaller, less storage it occupies in memory, therefore allowing an increase in the batch size, that translates to a faster and more stable convergence. Meanwhile, the image's resizing makes the inputs more homogeneous across different vendors. This results in the same structures being represented with consistent shapes and sizes across the B-scans from different vendors, resulting in a significantly simpler learning process.

It was seen in Figure 4.4 that the model predicted fluid masks in the choroid region. In Figure 4.6, it is seen that the models trained on vertical patches no longer predict fluid in the choroid region, despite inferring on the same B-scan. This means that, by feeding vertical patches to the model, it learns to not segment below the retina, in the choroid. This was not previously understood by the model when big patches were input, since these would frequently not capture

both the retinal layer and the choroid in the same patch, restraining the model from learning the relative position of the choroid and its relationship with the fluid prediction.

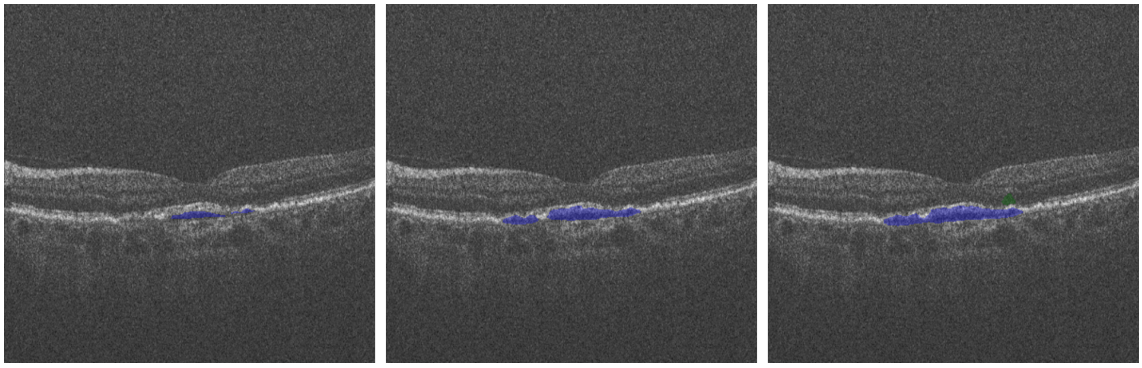

Figure 4.6: Predicted masks by the models trained on Run 16 (middle) and Run 20 (right) and their respective GT (left).

Afterwards, the use of seven and thirteen patches extracted from each B-scan was tested and the results are shown in Table 4.4. Two models were trained for each number of patches extracted, one validated on fold 2 and another validated on fold 3. The models were trained on 200 maximum epochs, with a 25 epoch patience applied after the first 100 epochs.

Table 4.4: Dice scores for every vendor and fluid for the runs done using seven (Runs 21 and 22) and thirteen (Runs 23 and 24) vertical patches extracted from each B-scan. Only two folds were used in order to understand the viability of using these numbers of patches, which is represented in the column "P". Runs 17 and 18 are shown here to make the comparison between the number of patches easier. The values shown in bold represent the best value for the models trained with the same number of patches.

Runs	**VF**	**P**	**Cirrus**			**Spectralis**			**Topcon**			**IRF**	**SRF**	**PED**	**Fluid**
			IRF	**SRF**	**PED**	**IRF**	**SRF**	**PED**	**IRF**	**SRF**	**PED**				
Run 17	2	4	0.522	0.784	**0.703**	**0.805**	0.765	**0.865**	0.828	0.879	0.835	0.677	0.813	**0.777**	0.684
Run 21	2	7	**0.556**	**0.837**	0.672	0.761	**0.853**	0.848	**0.829**	0.908	**0.858**	**0.685**	**0.864**	0.767	**0.681**
Run 23	2	13	0.500	0.762	0.635	0.672	0.790	0.773	0.819	**0.936**	0.805	0.639	0.826	0.718	0.637
Run 18	3	4	0.509	0.654	0.598	0.590	0.806	**0.755**	**0.777**	**0.787**	**0.792**	0.621	0.729	0.697	0.622
Run 22	3	7	**0.734**	**0.855**	0.836	**0.636**	**0.846**	0.689	0.686	0.781	0.731	**0.700**	**0.822**	**0.771**	**0.672**
Run 24	3	13	0.612	0.648	**0.882**	0.455	0.636	0.548	0.499	0.593	0.668	0.542	0.622	0.745	0.536

Training the models on more patches per B-scan, significantly increases the quantity of data that is seen by the model during an epoch. Therefore, each epoch takes much longer to complete, requiring more computation power, leading to often bottlenecks. However, since the model sees many more patches per epoch, the model reaches the validation loss minimum much earlier than in the models trained with four patches.

In Figure 4.7, it is possible to see a comparison between the training and validation losses of the models validated on fold 2, using four, seven, and thirteen vertical patches per B-scan. Despite following the same trends, the model learns significantly faster as the number of patches increases, resulting in a faster progression of the validation loss. As a result the models starts

to overfit much earlier and aggressively with larger number of patches, which is signaled by the increase in validation loss after reaching low values.

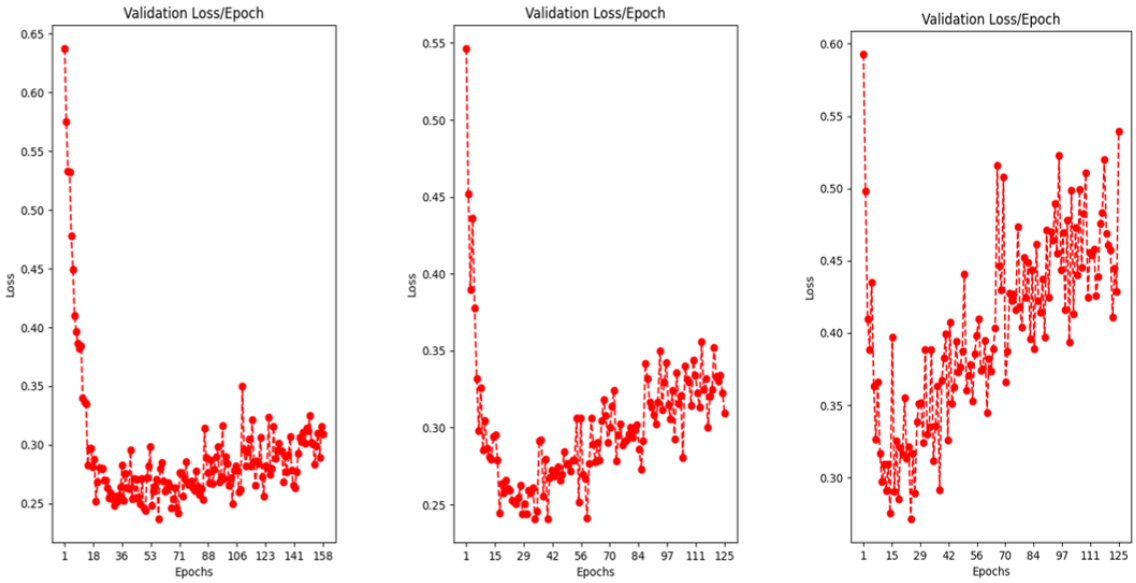

Figure 4.7: Training and validation losses for models validated on fold 2. The curves on the left are from the model trained on Run 17 with four patches, while the middle curves are from the model trained on Run 21, with seven patches. The curves on the right are from the model trained on Run 23, with thirteen patches.

When comparing the models' performances, by looking at Table 4.3, the models trained on thirteen patches are outperformed by those trained in fewer patches. These results suggest that the faster progression in validation loss for the model trained with thirteen patches inhibits its ability to converge to a solution as optimal as those achieved by models trained more slowly with fewer patches.

The performance differences between the models trained with four and seven patches are less pronounced than those when either is compared to the model trained with thirteen patches. However, the model trained with seven patches consistently outperforms the one trained with four in most metrics. In the few cases the model trained with four patches performs better, the other model still achieves comparable results. The reverse is not true: when the model trained with seven patches outperforms the model trained with four patches, the performance gap is significantly larger. For these reasons, the model trained with seven patches was considered superior.

Nevertheless, both models still make wrongful predictions, which do not respect the anatomic context of the image. For example, in Figure 4.8, the IRF fluid is predicted below the PED, which is not anatomically possible and there are no examples in the dataset that exhibit this relationship.

While this unexpected behavior could be attributed to the B-scan differing from the examples seen in training, the subsequent runs aimed to investigate whether this was caused by the rotation transformation. It was hypothesized that random rotations could alter the vertical order of fluid regions, potentially leading the model to learn incorrect patterns. Additionally, the padding that

is performed to fill the rotated areas, as illustrated in Figure 4.9, could also influence the model's performance.

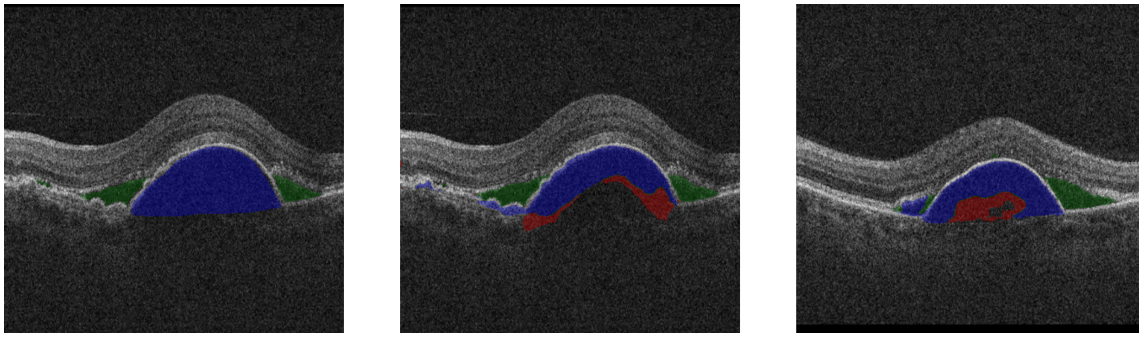

Figure 4.8: Segmentation errors in Cirrus B-scans. The GT fluid masks, seen on the left, show a large region of PED fluid. Meanwhile, the predictions made by the models trained in Run 21 and 23, which respectively correspond to the B-scans in the middle and right, classify the center of this region as IRF.

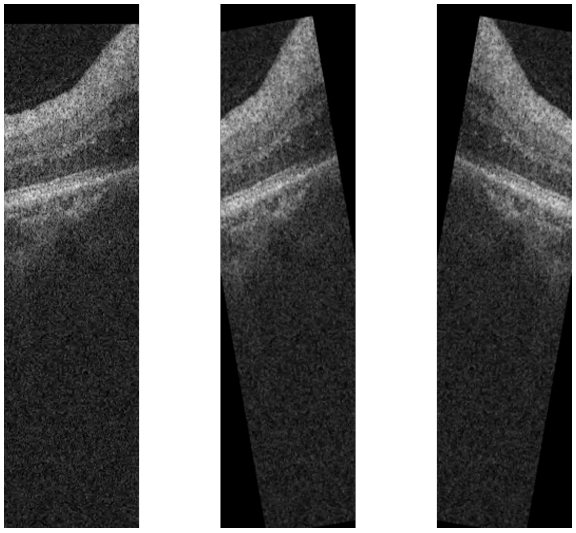

Figure 4.9: Transformations applied to a vertical patch of a Cirrus B-scan. The original vertical patch (left) can be rotated a maximum of 10°, a transformation that is shown in the middle image. The image on the right represents the combination of a 10° rotation and an horizontal flip. It is seen that the rotation transform pads a significant portion of the image, reducing the information in the input.

Table 4.5 shows the results obtained in the runs performed using only horizontal flipping as transformation, without performing rotations. In these runs, seven input patches were extracted from each B-scan. Due to the faster progression of the models trained using seven vertical patches per B-scan, the model was trained on a minimum of 25 epochs with a 25 epoch patience after. Since the validation minimum is usually attained early in the models trained with more patches, as seen in Figure 4.7, by setting such early stopping, the training would be much faster while maintaining the same performance.

Table 4.5: Dice scores for every vendor and fluid using seven vertical patches. The only transformation performed in these runs was horizontal flipping, without rotation. Runs 21 and 22 are also shown, promoting an easier comparison.

Runs	**VF**	**Cirrus**			**Spectralis**			**Topcon**			**IRF**	**SRF**	**PED**	**Fluid**
		IRF	**SRF**	**PED**	**IRF**	**SRF**	**PED**	**IRF**	**SRF**	**PED**				
Run 25	2	0.406	0.789	0.656	0.595	**0.805**	**0.786**	0.762	0.841	0.797	0.560	0.809	0.727	0.620
Run 26	3	0.589	0.775	0.553	0.586	0.803	0.766	0.769	**0.827**	0.755	0.654	0.799	0.664	0.617
Run 27	4	**0.797**	**0.845**	**0.827**	**0.734**	0.782	0.659	0.533	0.756	0.553	0.674	0.798	0.686	0.626
Run 28	0	0.677	0.842	0.699	0.681	0.799	0.732	**0.791**	0.846	**0.854**	**0.719**	**0.836**	**0.762**	**0.665**
Set 6	-	0.62	0.81	0.68	0.65	0.80	0.74	0.71	0.82	0.74	0.65	0.81	0.71	0.63
Run 21	2	0.556	0.837	0.672	0.761	0.853	0.848	0.829	0.908	0.858	0.685	0.864	0.767	0.681
Run 22	3	0.734	0.855	0.836	0.636	0.846	0.689	0.686	0.781	0.731	0.700	0.822	0.771	0.672

Removing the rotation from the transformations applied to the input data, significantly worsen the models performances. In Table 4.5, when comparing Run 21 to Run 25, which were validated on the same fold, the model trained in Run 21, in which rotation was applied, outperformed the model trained in Run 25 in every single metric. Following the same trend, the model trained in Run 26 only outperformed the model trained in Run 22 in four metrics: Spectralis PED and all the fluids in Topcon.

The differences in performance here exposed highlight the importance of applying transformations to the inputs, despite the mentioned loss of information associated with it. The introduction of variability to the input data significantly improved the model's generalization, preventing the model from overfitting so early on the training stage. In OCT, the rotation is particularly important since different images exhibit different orientations of the retinal layer. By introducing rotation to the inputs, the model becomes more robust to varied input data and less specific to the orientations seen in training. This transformation particularly enhances the performance of the models trained in B-scans where the retinal layer appears horizontal. In Figure 4.10, two B-scans that illustrate the variability of the retina orientation are shown.

Nevertheless, the lack of rotation in the training dataset mitigated the poor segmentation shown in Figure 4.8. As exhibited in Figure 4.11, the model that was trained without rotation does not predict as much IRF in the PED region as those trained with rotation, as shown in Figure 4.8. However, the model that was trained without rotation also predicted IRF in a region that none of the other models did, exemplifying why this model performed worse than those.

In order to find an equilibrium between the robustness that training a model with rotated patches brings and the respect for the vertical order and relationships of the fluids, new runs were performed with a maximum 5° rotation combined with the horizontal flipping. This rotation does not remove as much information from the patches as a 10° rotation. The results obtained in these runs are shown in Table 4.6.

When looking at the results in Table 4.6, the runs trained using a maximum 10° rotation attain a better Dice coefficient than those that are trained using 5°. However, it is important to also compare the performances only in the slices that have fluid, which are shown in Table 4.7.

The results shown in Table 4.7 depict comparable results between the models trained with rotations of 5° and 10° degrees. While the results still slightly favor those trained with 10° rotation,

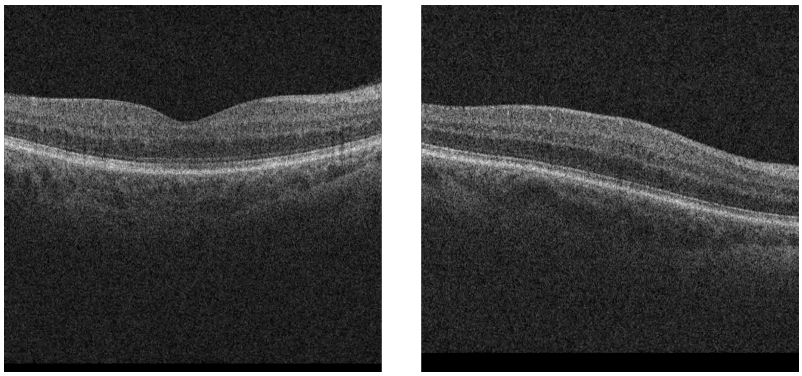

Figure 4.10: Example of two B-scans in which the retina is oriented differently. The retina present in the right image is significantly more inclined. Both B-scans were obtained using a Cirrus device and do not present fluid.

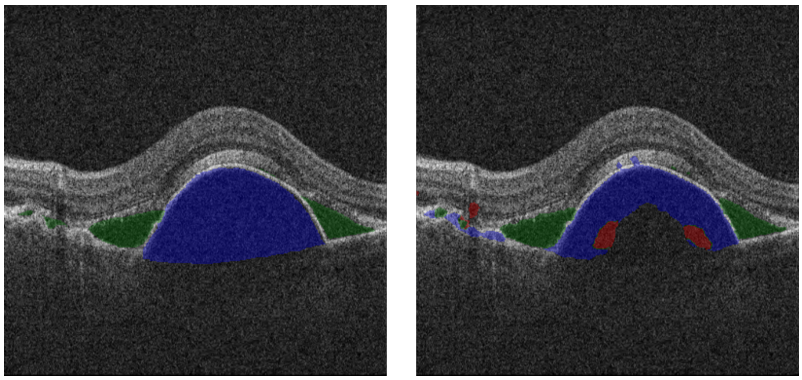

Figure 4.11: Segmentation errors in Cirrus B-scans when trained without random rotations, in the same B-scan as in Figure 4.8. In the left, the GT masks are shown, while in the right the predicted masks are exhibited.

Table 4.6: Dice scores for every vendor and fluid using seven vertical patches. In these runs, horizontal flipping and 5° rotation were used, instead of the usual 10°. The results obtained in Runs 21 and 22 are also shown, enabling a smoother comparison.

Runs	**VF**	**Cirrus**			**Spectralis**			**Topcon**			**IRF**	**SRF**	**PED**	**Fluid**
		IRF	**SRF**	**PED**	**IRF**	**SRF**	**PED**	**IRF**	**SRF**	**PED**				
Run 29	2	0.497	0.783	0.609	0.763	0.851	0.822	0.832	**0.904**	0.799	0.658	0.836	0.712	0.666
Run 30	3	0.475	0.747	0.613	0.656	0.846	0.757	**0.856**	0.869	**0.835**	0.646	0.809	0.720	0.646
Run 31	4	0.643	0.709	0.747	0.627	0.800	0.635	0.457	0.599	0.529	0.560	0.673	0.638	0.523
Run 32	0	**0.794**	**0.878**	**0.770**	**0.807**	**0.902**	**0.862**	0.796	0.842	0.830	**0.797**	**0.869**	**0.808**	**0.698**
Set 7	-	0.60	0.78	0.68	0.71	0.85	0.77	0.74	0.80	0.75	0.67	0.80	0.72	0.63
Run 21	2	0.556	0.837	0.672	0.761	0.853	0.848	0.829	0.908	0.858	0.685	0.864	0.767	0.681
Run 22	3	0.734	0.855	0.836	0.636	0.846	0.689	0.686	0.781	0.731	0.700	0.822	0.771	0.672

the difference between models is much smaller than it was in Table 4.6. The differences observed indicate that the Dice coefficient measured in the models trained with 10° rotation leverage on the high capability of detecting fluid.

This model is particularly good at detecting which slices have and do not have fluid. When the model does not detect fluid in a slice that does not have fluid, the Dice coefficient for that slice

Table 4.7: Dice scores for every vendor and fluid using seven vertical patches. The mean scores are calculated only for the B-scans that contain that fluid. For example, Cirrus IRF is the mean of the IRF Dice coefficient across all the Cirrus slices in the validation fold that contain IRF. The transformations utilized were the same as in Table 4.6.

Runs	**VF**	**Cirrus**			**Spectralis**			**Topcon**			**IRF**	**SRF**	**PED**	**Fluid**
		IRF	**SRF**	**PED**	**IRF**	**SRF**	**PED**	**IRF**	**SRF**	**PED**				
Run 29	2	0.499	0.730	**0.603**	0.596	0.854	0.583	**0.696**	0.629	0.577	0.575	0.731	0.588	0.645
Run 30	3	0.535	0.675	0.584	0.519	0.837	0.592	0.646	**0.735**	0.605	0.561	0.721	0.595	0.623
Run 31	4	**0.691**	0.637	0.602	0.619	**0.899**	0.623	0.506	0.651	**0.656**	0.603	0.709	**0.635**	**0.646**
Run 32	0	0.627	**0.815**	0.590	**0.705**	0.808	**0.646**	0.563	0.489	0.521	**0.627**	**0.761**	0.585	0.645
Set 7	-	0.61	0.74	0.59	0.65	0.85	0.65	0.58	0.62	0.55	0.60	0.75	0.59	0.64
Run 21	2	0.626	0.795	0.449	0.681	0.842	0.771	0.613	0.697	0.524	0.636	0.791	0.518	0.636
Run 22	3	0.720	0.687	0.554	0.653	0.914	0.543	0.569	0.679	0.615	0.647	0.738	0.583	0.651

is 1. Meanwhile, in case the slices do not have fluid but the model still detects fluid in them, the B-scan's Dice coefficient for that fluid is 0.

Due to the large quantity of slices that do not have fluid, which is larger than the slices that have fluid, a segmentation model can considerably improve its Dice performance by enhancing its capability of detecting fluid. Despite the model trained with 5° performing worse at detecting fluid, most of these wrongful predictions are just a small quantity of sparse pixels (less than 100 pixels in each 496×512 image).

It is also of interest to look into the predictions made by each model. While the Dice coefficient is a good metric to represent the models performance, some relevant segmentation characteristics can not be expressed by this measurement.

When looking at the predictions made by both models, the model trained with 5° rotations attains performances that are closer to the GT. An example of this can be seen in Figure 4.12, where a comparison between the predictions made by different models are shown. In this image, it is seen that the masks predicted (middle image) by the model trained with 10° rotation segments all the PED region, but also segments much more PED overall. Meanwhile, the model trained with 5° rotation also identifies the PED region correctly, but segments less PED in the image (right image). The segmentation of IRF and SRF are really similar between both models.

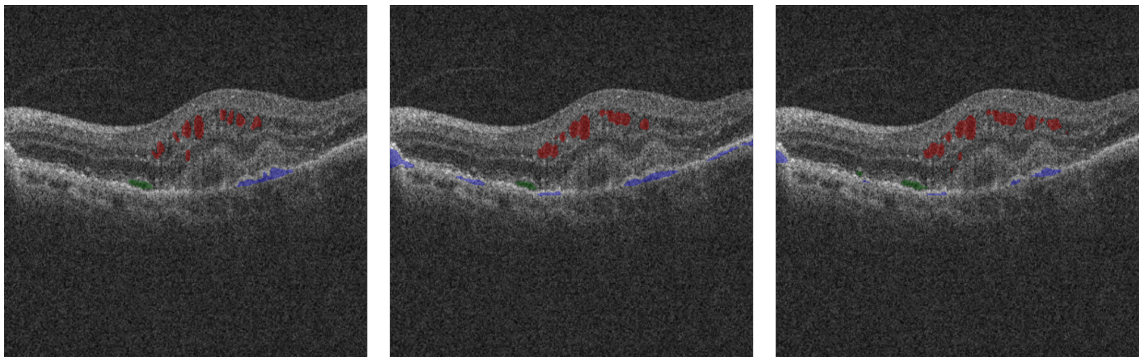

Figure 4.12: Predictions by the models trained using 5° (middle image) and 10° (right image) rotations, compared with the respective GT (left image).

This is an example of how the model trained with 5° rotations tends to be more conservative in the fluid segmentation, despite not being as good at identifying fluid in the B-scans. For this reason, it was expected that this model would perform better on unseen data than the model trained with 10° which tends to oversegment. Therefore, the maximum rotation selected for further experiments was 5°.

Following the experiments that applied a 5° rotation to the seven vertical patches extracted from each B-scan, additional runs made using the same rotation but with only four patches extracted from each B-scan instead. The objective of these runs was to verify if model still performs better on seven patches than on four patches when the applied rotation is changed from 10° to 5°. The results are shown in Table 4.8.

Table 4.8: Dice scores for every vendor and fluid using four vertical patches. The rotation applied in these runs was of 5°, instead of the previously used 10°. "Set 7" and the best performing run in this set (Run 32) are also shown, promoting an easier comparison between using four and seven patches. The underlined values correspond to the best performances between the models trained in Run 32 and Run 33.

Runs	**VF**	**Cirrus**			**Spectralis**			**Topcon**			**IRF**	**SRF**	**PED**	**Fluid**
		IRF	**SRF**	**PED**	**IRF**	**SRF**	**PED**	**IRF**	**SRF**	**PED**				
Run 33	2	0.598	**0.824**	0.719	**0.805**	0.873	**0.887**	**0.848**	**0.947**	**0.863**	**0.720**	**0.874**	**0.798**	**0.733**
Run 34	3	0.298	0.701	0.436	0.442	0.785	0.720	0.717	0.814	0.746	0.477	0.757	0.599	0.530
Run 35	4	0.670	0.796	**0.850**	0.531	0.825	0.701	0.509	0.703	0.674	0.582	0.759	0.754	0.578
Run 36	0	**0.695**	0.816	0.737	0.761	**0.882**	0.838	0.728	0.867	0.775	0.719	0.846	0.769	0.635
Set 8	-	0.57	**0.78**	**0.69**	0.63	0.84	**0.79**	0.70	**0.83**	**0.76**	0.62	**0.81**	**0.73**	0.62
Run 32	0	0.794	0.878	0.770	0.807	0.902	0.862	0.796	0.842	0.830	0.797	0.869	0.808	0.698
Set 7	-	**0.60**	**0.78**	0.68	**0.71**	**0.85**	0.77	**0.74**	0.80	0.75	**0.67**	0.80	0.72	**0.63**

Contrasting with the results shown in Table 4.3 and Table 4.4, which show a significant improvement when using seven vertical patches over four, the results exhibited in Table 4.8 present a similar performance between the models trained on four and seven vertical patches. The main performance differences are in the IRF segmentation, where the difference is more significant depending on the number of patches used, favoring the models trained with seven patches.

However, it is necessary to select a single segmentation model to perform inference in the fold that was reserved. The best performing model from those trained with seven vertical patches obtained from each B-scan and a maximum rotation of 5° is the model trained in Run 32. Meanwhile, the best performing model from "Set 8" is Run 33. In Table 4.8, the underlined values correspond to the best performance attained between Run 32 and 33.

In most metrics, the model trained on Run 32 performs better than the model trained on Run 33. Nevertheless, both performances are really similar in most metrics, with the only significant difference being the IRF segmentation, which is superior in Run 32. It is important to note that the models were nor trained nor validated on the same data and an assumption that this would directly translate to the performance in unseen data was made. However, it is possible that the data in which one model was trained prepares it better for the data on the reserved fold. Therefore, the U-Net model selected to represent the multi-class segmentation approach was the one trained in Run 32. This model subsequently performed the inference in the unseen fold (fold 1) and its results are

shown in Table 4.9. In this table, the rows correspond to the model performances on different sets of slices. In the first row the performance across all slices is considered. In the second row only the slices with the fluid indicated in the column are quantified, while in the third row only those that do not present that fluid are considered.

Table 4.9: Dice scores for every vendor and fluid in the reserved fold (fold 1). The first row reports the mean Dice across all slices. The second row shows the mean for slices containing the fluid specified in the column, while the third shows the mean for the slices without that fluid.

Runs	**Slices**	**VF**	**Cirrus**			**Spectralis**			**Topcon**			**IRF**	**SRF**	**PED**	**Fluid**
			IRF	**SRF**	**PED**	**IRF**	**SRF**	**PED**	**IRF**	**SRF**	**PED**				
Run 32	All	1	0.852	0.918	0.934	0.645	0.821	0.852	0.818	0.929	0.755	0.799	0.905	0.841	0.742
	Fluid	1	0.654	0.799	0.767	0.645	0.687	0.703	0.624	0.744	0.588	0.641	0.756	0.716	0.702
	No Fluid	1	0.941	0.972	0.978	0.646	0.889	0.885	0.884	0.960	0.766	0.873	0.952	0.862	0.786

The results shown in the table indicate that the model generalized well. The performance obtained in the reserved fold, whose data had not been used in training or validation, reveals the model's robustness across different fluids and vendors. Since the model performs equally well across the three vendors, a good cross-device generalization has been achieved. Therefore, the model is not dependent on the image quality of the B-scans to produce accurate predictions.

The model behaved differently depending on the fluid expected to segment. The Dice coefficient was considerably larger in SRF and PED than in IRF. This is associated with the larger variety of IRF shape and position in the retina. IRF regions are usually irregular in shape, smaller, and less well-defined than the other fluids, often appearing as small structures. This fluid is also located between the retinal layers, making it harder for the model to distinguish from the surrounding tissue and often leading to labeling ambiguity. Still, the model attained a better performance in fluid segmentation than those seen in Table 4.7, where only fluid is considered.

When considering all the slices, the Dice coefficient improves significantly, when compared to its performance where only slices with fluid are considered. At the same time, the Dice coefficient obtained in the slices without fluid is also better than the fluid segmentation. This reflects the greater difficulty of accurately segmenting the fluids, when compared to the task of detecting their presence. Since most slices in the dataset do not present fluid, an accurate detection of it can significantly increase the Dice mean across all slices, as explained previously.

Overall, the performance obtained when validating in this unseen fold is significantly better than any other validation performance in previously seen in Table 4.6. This indicates that the model did not overfit on training data, benefiting from the diversity introduced in training both through varying images and the transformations applied to them. This performance also benefits from selecting the best performing model in cross-validation, which allows the choice of a robust model.

In Figures 4.13, 4.14, and 4.15, some examples of predictions performed by the model in the unseen data are shown. In the IRF segmentations, it is seen that the model can not correctly identify the retinal tissue that separates the fluid regions and connects nearby fluid regions. The PED segmentation is marked by accurate delimitation of the large PED regions, with some predictions

made in B-scans without PED. Lastly, SRF is the fluid in which the prediction segmentation more resembles the GT.

The performance seen by this model in the reserved fold was subsequently compared with the best performing models from the following experiment.

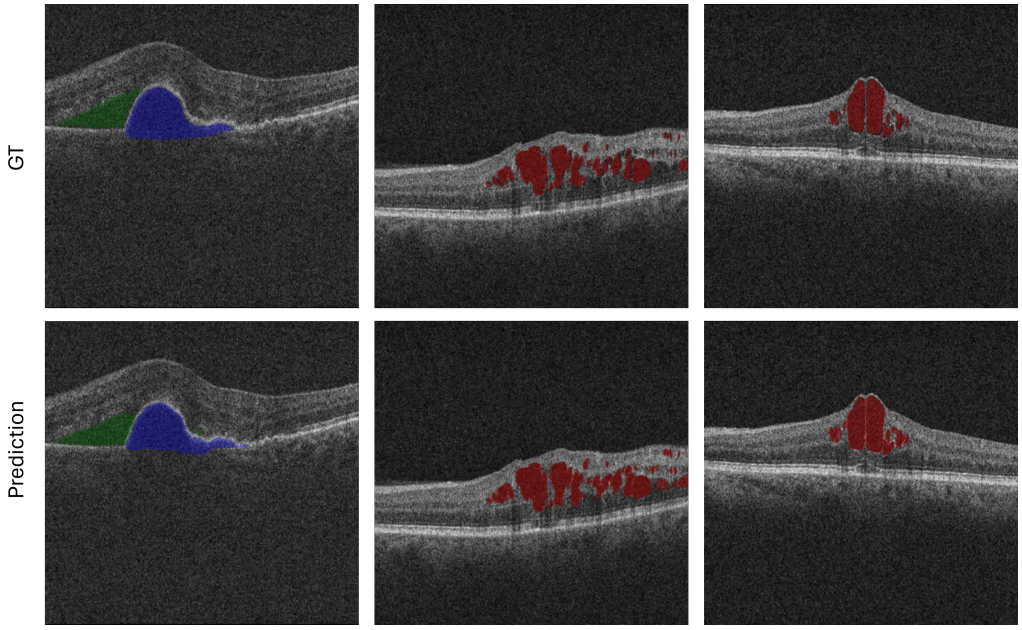

Figure 4.13: Predictions by the model trained on Run 32 in unseen Cirrus volumes of fold 1.

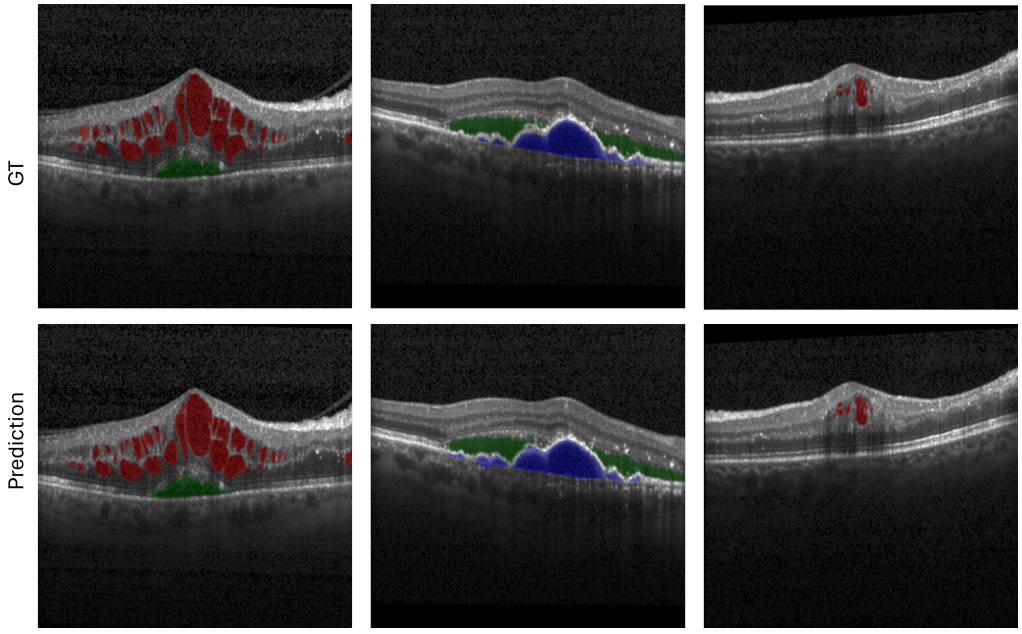

Figure 4.14: Predictions by the model trained on Run 32 in unseen Spectralis volumes of fold 1.

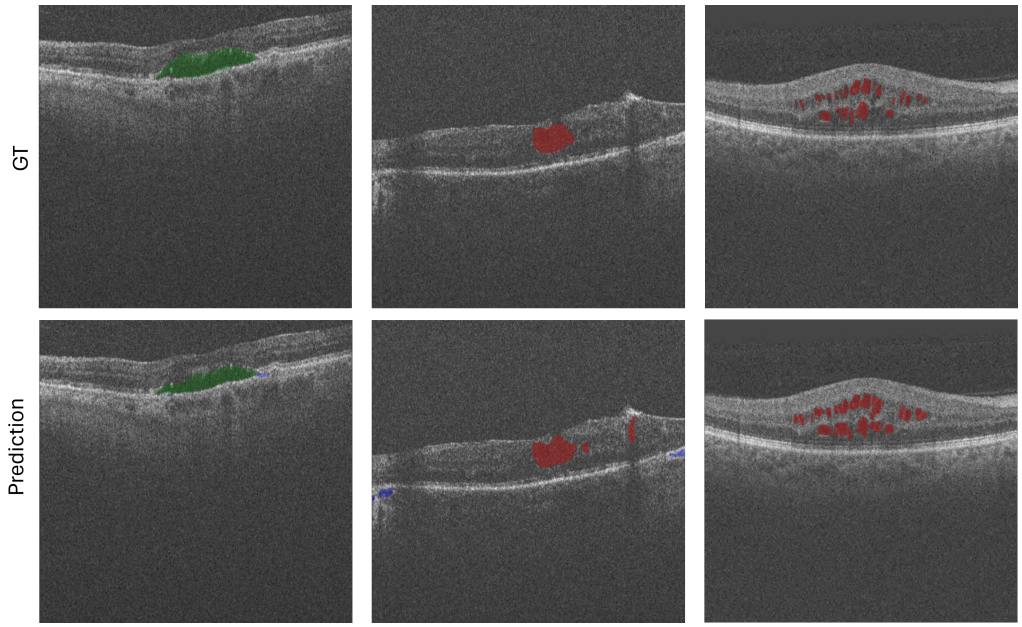

Figure 4.15: Predictions by the model trained on Run 32 in unseen Topcon volumes of fold 1.

4.1.2 Experiment 2

The results shown in this subsection correspond to the use of three separate U-Nets, one for each fluid, to perform multi-class fluid segmentation. In Experiment 2.1, the loss function was the same as in Experiment 1, while in Experiment 2.2 the loss function was changed to the weighted cross-entropy. The results in these experiments were compared with each other.

4.1.2.1 Experiment 2.1

In this experiment, for each fluid, eight models were trained. One model was trained for each fold in the multi-class 5-fold split and then one model was trained for each fold in a fluid-specific 5-fold split.

Table 4.10 illustrates the performance of the models trained to segment IRF, with the upper block corresponding to the models trained on the multi-class split while the second block shows the performances for the models trained on the split specifically made for IRF segmentation.

The results shown in this table follow a well defined trend: the IRF segmentation in the slices which contain this fluid performs similarly to the segmentation predicted by multi-class models, while the performance when considering all the slices is significantly worse.

The Dice obtained in segmentation in the slices with IRF is really similar to the values obtained when training multi-class models, as the difference between the mean of the runs performed is around 0.01 for all vendors. Changing the split in which the models were trained barely made a difference in the performances' average. However, while the segmentation in slices with IRF almost did not change, the mean performance in all the slices saw a significant improvement in Cirrus and a slight increase in Spectralis, combined with a significant decrease in Topcon.

Table 4.10: IRF Dice scores for every vendor. Runs 37 to 40 utilize the multi-class 5-fold split from Experiment 1, while the Runs 41 to 44 use the fluid-specific 5-fold split. The results are presented both for every slice and for the slices which contain IRF.

Runs	**VF**	**Cirrus**		**Spectralis**		**Topcon**		**IRF**	
		All	**Fluid**	**All**	**Fluid**	**All**	**Fluid**	**All**	**Fluid**
Run 37	2	0.478	0.634	0.695	0.669	0.735	0.569	0.604	0.625
Run 38	3	0.536	0.452	0.514	0.588	**0.820**	**0.706**	0.637	0.553
Run 39	4	0.545	**0.683**	0.617	0.670	0.540	0.484	0.552	0.600
Run 40	0	0.576	0.619	0.695	0.682	0.661	0.599	0.628	0.628
Set 9	-	0.53	0.60	0.63	0.65	0.69	**0.59**	0.61	0.60
Run 41	2	**0.654**	0.663	0.541	0.606	0.648	0.619	0.632	0.635
Run 42	3	0.507	0.641	**0.741**	0.670	0.775	0.675	**0.649**	**0.656**
Run 43	4	0.583	0.516	0.689	0.660	0.310	0.451	0.502	0.546
Run 44	0	0.618	0.566	0.644	**0.686**	0.588	0.600	0.611	0.594
Set 10	-	0.59	0.60	0.65	**0.66**	0.58	**0.59**	0.60	**0.61**
Set 7	-	**0.60**	**0.61**	**0.71**	0.65	**0.74**	0.58	**0.67**	0.60

The reason Topcon behaves differently is because, overall, the volumes obtained with devices from this vendor present less fluid than the volumes obtained with the other devices. Therefore, the model has to analyze the Topcon images more carefully and be more attentive of the smaller details, as the fluid regions are not as big as those seen in the other vendors. This leads to the model being better at separating the slices with fluid from those that do not present fluid. As seen in Experiment 1, an effective detection of fluid can significantly improve the model's Dice coefficient. The implications this has in model performance's are also visible in the multi-class segmentation results, as shown in the performance differences in B-scans from Topcon between Table 4.6 and Table 4.7.

When comparing the results across all slices between these binary models with the multi-class, it is seen a significant decrease in Dice coefficient in the models tasked to segment exclusively IRF. There are two main causes for this decrease, with the first being an increment in the number of voxels labeled as background. As the model attempts to exclusively segment IRF, all the voxels that are not labeled as IRF, are classified as background. This means that the regions of SRF and PED are also interpreted as background. The images in the dataset are predominantly composed of non-fluid voxels, but, by re-assigning the other fluids' voxels in the retina as background, the proportion of fluid voxels in the image, and particularly in the retina, becomes even smaller. As the number of background voxels increase, the more chances the model has of mislabeling them as fluid.

The increased number of background voxels is further impacted with the lack of inter-class competition. This inter-class competition is particularly helpful in the segmentation of other ambiguous regions such as other fluids, which, in binary segmentation, is not contested by other classes. This does not incentive the model to learn the anatomical characteristics of the retina that are key to identifying the region corresponding to each fluid, often leading to over-segmentation.

This translates to a decrease in the IRF segmentation model's performance, particularly in the slices that do not present this fluid. The same performance decrease is also seen in SRF and PED, as shown further, due to the same reasons.

Table 4.11 shows the performance of the models trained on the same conditions as those trained to segment IRF, but now with the goal of segmenting SRF. Similarly, on the second block of this table, the models were trained using a 5-fold split specifically made to evenly distribute the OCT volumes according to the quantity of SRF in them.

Table 4.11: SRF Dice scores for every vendor. Runs 45 to 48 utilize the multi-class 5-fold split from Experiment 1, while the Runs 49 to 52 use the fluid-specific 5-fold split. The results are presented both for every slice and for the slices which present SRF.

Runs	**VF**	**Cirrus**		**Spectralis**		**Topcon**		**SRF**	
		All	**Fluid**	**All**	**Fluid**	**All**	**Fluid**	**All**	**Fluid**
Run 45	2	**0.866**	0.784	0.882	0.826	**0.956**	0.645	**0.899**	0.774
Run 46	3	0.599	0.574	0.741	0.843	0.820	0.737	0.705	0.666
Run 47	4	0.769	0.645	**0.874**	0.873	0.845	0.689	0.816	0.728
Run 48	0	0.667	0.795	0.765	0.776	0.773	0.643	0.723	0.765
Set 11	-	0.73	0.70	0.82	0.83	**0.85**	0.68	0.79	0.73
Run 49	2	0.509	0.758	0.617	0.793	0.759	0.607	0.613	0.745
Run 50	3	0.743	0.733	0.776	**0.885**	0.901	0.801	0.807	0.779
Run 51	4	0.720	0.722	0.864	0.875	0.816	**0.803**	0.781	0.809
Run 52	0	0.775	**0.872**	0.810	0.826	0.879	0.676	0.819	**0.826**
Set 12	-	0.69	**0.77**	0.77	0.84	0.84	**0.72**	0.75	**0.79**
Set 7	-	**0.78**	0.74	**0.85**	**0.85**	0.80	0.62	**0.80**	0.75

As seen in this table, the performances in the segmentation of SRF in slices that contain this fluid have attained improved results when compared to those obtained using the multi-class model, especially when using the volume's distribution specific for SRF. In these conditions, the binary model significantly outperformed the multi-class model in the segmentation of slices with SRF from Cirrus and Topcon, while performing almost equally in the Spectralis slices.

When looking at the models trained on the SRF 5-fold split, similarly to what happened in the IRF model, a significant performance decrease is seen when all the slices are considered. In these metrics, the model is outperformed by the multi-class model in every column except in the Topcon slices, where the performance increases. Similar to what was seen in the IRF segmentation, the performances in the Topcon volumes are not affected as much when performing binary segmentation due to the low quantity of fluid present in the OCT scans from this vendor.

The results obtained using the multi-class split follow the same trend in Topcon. However, in Cirrus, it is seen a small increase in the Dice coefficient when considering all the slices. This is caused by the not so even volumes distribution, resulting in some folds having easier volumes for fluid detection, which have less fluid. This reflects in the high variance seen in the performance considering all slices in Cirrus. Contrasting, when the data is split equally, the dispersion is less pronounced, with only one outlier in that device (Run 49). Despite being more noticeable in

Cirrus, Spectralis presents the same issue, where the results when using the multi-class split differ from what is expected and have sparser values.

Overall, the Dice coefficient obtained in the SRF binary segmentation is better than the Dice obtained in the IRF. This same trend is also seen in Experiment 1, where SRF is the best performing fluid.

The runs in which a binary segmentation U-Net was trained to segment PED are shown in Table 4.12, where the behavior is comparable with the ones seen in IRF and SRF.

Table 4.12: PED Dice scores for every vendor. Runs 53 to 56 utilize the multi-class 5-fold split from Experiment 1, while the Runs 57 to 60 use the fluid-specific 5-fold split. The results are presented both for every slice and for the slices which present PED.

Runs	**VF**	**Cirrus**		**Spectralis**		**Topcon**		**PED**	
		All	**Fluid**	**All**	**Fluid**	**All**	**Fluid**	**All**	**Fluid**
Run 53	2	0.305	0.482	0.555	0.623	0.626	0.552	0.459	0.522
Run 54	3	0.447	0.508	0.437	0.628	**0.769**	0.623	0.563	0.583
Run 55	4	0.368	0.594	0.346	0.566	0.328	0.618	0.348	0.600
Run 56	0	0.484	0.547	0.326	0.639	0.697	0.460	0.534	0.545
Set 13	-	0.40	0.53	0.42	0.61	0.61	0.56	0.48	0.56
Run 57	2	0.357	0.561	0.304	0.720	0.456	0.593	0.381	0.594
Run 58	3	0.232	0.581	0.305	0.574	0.361	0.568	0.292	0.574
Run 59	4	0.456	**0.648**	0.345	0.601	0.588	**0.775**	0.499	**0.703**
Run 60	0	**0.515**	0.620	**0.558**	**0.769**	0.672	0.529	**0.580**	0.630
Set 14	-	0.39	**0.60**	0.38	**0.67**	0.52	**0.62**	0.44	**0.63**
Set 7	-	**0.68**	0.59	**0.77**	0.65	**0.75**	0.55	**0.72**	0.59

In the binary segmentation of PED, the Dice coefficients obtained in the segmentation of slices with this fluid attained slightly better results when compared with those obtained in Experiment 1. However, this is only true when using the 5-fold split specifically made for this fluid. As seen in the previous experiments, the performance declines when considering all the slices.

In the context of PED, it is important to understand the distribution of the fluid across the OCT volumes. While in IRF and SRF all volumes contain at least one of these fluids, in varying quantities, PED appears only in a smaller number of volumes and, in general, in large quantities. This makes the fair division across folds harder, since in each fold the B-scans that compose it are either with a large deformation caused by PED or, in most cases, without PED at all. This happens because most of the OCT volumes that present PED are in severe cases, where large quantities of all fluids appear. This large intra-class variability significantly affects the performance as it increases the difficulty of segmentation, leading to multiple occurrences of oversegmentation.

This uneven distribution contributes to a poor performance when using all slices. It was already exposed that the models attained better fluid detection when dealing with smaller quantities of fluid, which resulted in a better detection performance in Topcon, where IRF and SRF appear in smaller areas. Now, when dealing with PED, the same trend is seen. However, in the other vendors, where PED appears in a larger quantity, the regions with this fluid generally occupy either larger

areas or do not appear at all. This confuses the models, often leading to a segmentation of PED in slices where it is not present. This issue did not affect multi-class segmentation as the model capability of understanding the retina anatomy was more developed, as it was required to attribute the correct label to each voxel, which motivated the association between PED regions and the deformations seen in the retina.

This translates to a good performance in the slices with fluid, but poor performance when all slices are considered. When the data partition is made specifically for PED, the quantity of fluid is fair in training and validation and the performances tend to what is expected: a significant decrease when all slices are considered, when compared to the segmentation in the slices with fluid, where the performance is decent. When using the multi-class segmentation split, fluid is not as evenly distributed, resulting in folds with more slices with PED and other folds with more slices without this fluid. In this case, and particularly in Topcon, models trained in folds that contain less fluid perform better at detecting fluid. For example, in Run 54, the models are trained on Topcon volumes with lower quantities of PED while fold 4 contains larger amounts, leading to correct detection of fluid in B-scans without any voxels labeled with this fluid. Inversely, in Run 55, the validation fold is composed of Topcon volumes with small amounts of PED, but trained on volumes with larger quantities. This results in oversegmentation in validation, thus attaining small Dice values when considering all the slices, but a decent score in the slices that have PED.

In order to compare the results from this Experiment to the ones obtained in Experiment 1, three models had to be selected: one for IRF, one for SRF, and one for PED. Since the results obtained in Experiment 2.2 were much worse than the results reached in this experiment, the models selected for the inference in the unseen fold were selected from the runs performed in Experiment 2.1.

From the models trained for IRF segmentation, ranging from Run 37 to Run 44, the model which obtained the best performance was the one trained on Run 42, as seen in Table 4.10. This model reached the best average performance in the segmentation of IRF both in all slices and when only slices with IRF were considered. In the other metrics, this model's performance were consistently similar to the best performing model in each metric. When compared to the average model from "Set 7", this model reached better performances in most metrics.

The model selected to infer the SRF masks in the unseen fold was the model trained on Run 52. This model was the best performing SRF model in fluid segmentation in slices where SRF is present. In the other metrics, it performed closely to the best models, surpassing the average model from "Set 7" in most metrics.

In PED, the selected model was the one trained on Run 59. This model was the best model in PED segmentation in slices that contain this fluid, with a significant difference from the other models. When considering all the slices, the model was the second best, but with a weak performance. Regarding the slices with fluid, this model outperformed the average from "Set 7" in all vendors. However, similar to all the other models trained for the binary segmentation of PED, it achieved a significantly worse performance when compared to the ones obtained in multi-class.

When merging the masks from three independent binary segmentation models with the goal of

outputting a multi-class segmentation, overlaps may occur, where multiple models assign different fluid labels to the same voxel. To resolve this conflict, two alternatives were explored in this fold: a priority-based approach, where the SRF takes precedence over IRF, which in turn takes precedence over PED; and a probability-based approach, where the predicted probabilities for the voxel are compared, attributing the label with highest predicted probability to it.

The results obtained using the three selected models are shown in Table 4.13, where the first block corresponds to the results when using the priority-based merging approach and the second block presents the results when using the probability-based approach.

Table 4.13: Dice scores for every vendor and fluid in the reserved fold (fold 1), using the models from Runs 42, for IRF, 52, for SRF, and 59, for PED. The first block corresponds to the results when using the priority rule merging strategy, while the second block is associated with the results when using highest probability as the merging strategy. In each block, the first row reports the mean Dice across all slices. The second row shows the mean for slices containing the fluid specified in the column, while the third shows the mean for the slices without that fluid. The values marked in bold are the best values when corresponding rows are compared, so that, for example, the row "All" in the first block is compared to the row of same name in the second.

Runs	**Slices**	**VF**	**Cirrus**			**Spectralis**			**Topcon**			**IRF**	**SRF**	**PED**	**Fluid**
			IRF	**SRF**	**PED**	**IRF**	**SRF**	**PED**	**IRF**	**SRF**	**PED**				
Runs 42, 52, and 59	All	1	0.693	0.776	0.425	0.587	0.623	0.177	0.808	0.858	**0.531**	0.723	0.783	0.425	**0.544**
	Fluid	1	0.687	**0.670**	0.573	**0.660**	**0.659**	0.632	**0.634**	**0.560**	**0.484**	**0.661**	**0.640**	0.570	**0.639**
	No Fluid	1	0.695	0.824	**0.387**	**0.520**	0.605	**0.075**	0.867	0.907	**0.534**	0.752	0.830	**0.402**	**0.440**
Runs 42, 52, and 59	All	1	**0.697**	**0.787**	**0.428**	**0.628**	**0.631**	**0.178**	**0.810**	**0.864**	**0.531**	**0.733**	**0.792**	**0.427**	**0.544**
	Fluid	1	**0.688**	0.663	**0.586**	**0.660**	0.646	**0.636**	0.632	0.537	**0.484**	**0.661**	0.627	**0.578**	**0.639**
	No Fluid	1	**0.701**	**0.844**	**0.387**	**0.520**	**0.623**	**0.075**	**0.870**	**0.917**	**0.534**	**0.766**	**0.844**	**0.402**	**0.440**

The performances of the models shown in this table, indicate that the best merging strategy is the probability-based. In the slices that contain the evaluated fluid, the priority-based approach performs better in SRF, since it is the fluid of highest priority. Regarding IRF, in the slices where this fluid is present, the performance is equal for both approaches, while in PED the probability-based approach is better.

However, when considering all the slices, the approach based on the probability of each model is significantly better. This is tied to the better performance in the slices that do not contain at least on type of fluid. In these slices, this approach prevents the segmentation of the fluids which are not present in the slice. These oversegmentations can be wrongfully predicted in the region of other fluids. However, since the models responsible for correctly segmenting this region are more confident than the models that are wrongfully segmenting it, the labels are attributed to the correct class.

It is important to note that the "Fluid" column does not change when using different merging approaches. This is because the region of fluid segmented stays the same, since the only changes performed are in voxels contested by multiple fluids, and if all the fluids belonged to a single class, as considered in this column, no changes would be made.

The overall segmentation performance was comparable to the values obtained in the validation folds, during training. In IRF, the model trained in Run 42 reaches better results in this unseen fold

than in the fold in which it was originally validated. This is true when considering all the slices, just those with IRF, or the slices without IRF, which signals the good generalization by the model.

In the segmentation of SRF, the model did not perform as good in this fold. The fluid detection was better, seen by the significant increase in Dice coefficient when considering all the slices instead of just those with fluid, which opposes the results obtained in the validation in Run 52, where the performance decreases in this situation. However, the segmentation worsened significantly in the slices with SRF, as the Dice coefficient dropped from 0.819 in its original validation fold to 0.627. Nevertheless, when considering all the slices, the Dice coefficients are still similar, largely due to the improved Dice in the slices without SRF.

When segmenting IRF and SRF, the models benefited from a good generalization in fluid detection task, which surpassed the performances originally obtained in validation. However, in PED, where the models were the most sensitive in this matter, the performance in this task was worse. It is the only fluid in which the Dice is worse when considering all the slices than when considering just the slices with the fluid. This performance is tied to the model selected for PED segmentation. As seen in Table 4.12, the model trained in Run 59 already presents a subpar performance in fluid detection, seen by the decrease in Dice coefficient when considering all the slices instead of those that contain PED. Therefore, the poor detection in unseen images is not unexpected. Note that the model selected detects fluid worse in the Spectralis volumes, similar to what was seen in the model validation. Similarly, the vendor in which the best performance is attained is the Topcon, where the model also performed best during validation. Moreover, all the PED models were expected to generalize worse in the unseen fold in the fluid detection task, due to the reasons previously presented that justify the lack of robustness in these models, such as the heterogeneous distribution of PED in OCT volumes.

4.1.2.2 Experiment 2.2

In this experiment, the loss function used in the training of the segmentation models was changed from a combination of a weighted cross-entropy and Dice coefficient of the foreground to the weighted cross-entropy. The results are shown in Table 4.14, where only two runs were performed for the segmentation of IRF, one in each 5-fold split.

The results seen in the table are much worse than those obtained in the same conditions with the previous loss. In the slices with IRF, the segmentation was poor, but with a performance almost comparable to the values obtained in Run 40 and 43.

However, what really worsens the overall performance is fluid detection. All the Dice coefficients in the slices with no fluid were inferior than 0.05, while most of the models studied in Experiment 2.1 reached values superior than 0.5.

The reason these values were so low is related with the loss function used. By keeping the Dice loss as a component of the training loss, the model is highly penalized when predicting fluid in an image which does not have fluid. Whenever any fluid is predicted in such image, the Dice coefficient is 0. This motivates the model to carefully predict voxels in regions of the image where it highly believes there is fluid.

Table 4.14: IRF Dice scores for every vendor, using the binary segmentation U-Net for IRF. The loss function used in Runs 61 and 62 is the weighted cross-entropy. This model was trained on two different validation folds: validation fold 0 from the multi-class 5-fold split, and validation fold 4 from the 5-fold split specific for IRF. These results can be compared, respectively, to those obtained in Runs 40 and 43. The metrics are compared between rows with the same name, so that the values in bold are the best across the four experiments.

Runs	Slices	VF	Cirrus	Spectralis	Topcon	IRF
Run 61	All	0	0.088	0.164	0.121	0.113
	Fluid	0	0.320	0.390	0.472	0.378
	No Fluid	0	0.002	0.044	0.041	0.024
Run 62	All	4	0.130	0.250	0.142	0.151
	Fluid	4	0.399	0.531	0.549	0.466
	No Fluid	4	0.000	0.052	0.053	0.030
Run 40	All	0	0.576	**0.695**	**0.661**	**0.628**
	Fluid	0	**0.619**	**0.682**	**0.599**	**0.628**
	No Fluid	0	0.555	0.705	**0.686**	**0.628**
Run 43	All	4	**0.583**	0.689	0.310	0.502
	Fluid	4	0.516	0.660	0.451	0.546
	No Fluid	4	**0.601**	**0.719**	0.272	0.486

When removing the Dice component from the loss, the wrongful predictions are only penalized by the cross-entropy component. This component does not penalize as much the predictions of fluid in empty images. The cross-entropy of each B-scan is calculated adjusted to the number of voxels belonging to a class in the image, as shown in 3 Methods. Whenever a background voxel is wrongfully classified as fluid, the resulting penalty is really small, due to the large quantity of background voxels in the mask. Therefore, the incentive to not segment fluid in images which are solely composed of background is much smaller than it is when the Dice is considered.

The cross-entropy is much more useful when regularizing the wrong labeling, especially in fluid. Since this loss component is adjusted to the representation of each class in the image, wrongfully labeling a fluid region as another fluid or background brings a large penalty to the model. This is why it is combined with the Dice loss in the original loss: while the Dice ensures that the correct areas of fluid are segmented, the cross-entropy makes sure that the correct labels are assigned to each voxel. For this reason, most of the segmentation errors that are seen in the predictions made by the models trained in Run 61 and 62 are oversegmentations, and not as many incorrect labelings.

4.1.2.3 Experiment 1 and Experiment 2 Comparison

4.2 Intermediate Slice Synthesis

4.2.1 Experiment 3

4.2.2 Experiment 4

4.3 Fluid Volume Estimation

4.3.1 Experiment 5

4.3.2 Experiment 6

References

[1] F. Hutmacher, "Why Is There So Much More Research on Vision Than on Any Other Sensory Modality?" *Frontiers in Psychology*, vol. 10, 2019, ISSN: 1664-1078. DOI: 10.3389/fpsyg.2019.02246.

[2] H. Bogunović, W.-D. Vogl, S. M. Waldstein, and U. Schmidt-Erfurth, "Chapter 14 - OCT fluid detection and quantification," in E. Trucco, T. MacGillivray, and Y. Xu, Eds. Academic Press, 2019, pp. 273–298, ISBN: 978-0-08-102816-2. DOI: 10.1016/B978-0-08-102816-2.00015-0.

[3] L. S. Lim, P. Mitchell, J. M. Seddon, F. G. Holz, and T. Y. Wong, "Age-related macular degeneration," *The Lancet*, vol. 379, pp. 1728–1738, 9827 2012, ISSN: 0140-6736. DOI: 10.1016/S0140-6736(12)60282-7.

[4] P. Mitchell, G. Liew, B. Gopinath, and T. Y. Wong, "Age-related macular degeneration," *The Lancet*, vol. 392, pp. 1147–1159, 10153 Sep. 2018, ISSN: 0140-6736. DOI: 10.1016/S0140-6736(18)31550-2.

[5] O. Musat et al., "Diabetic Macular Edema," *Rom J Ophthalmol*, vol. 59, pp. 133–136, 3 Jul. 2015.

[6] N. Bhagat, R. A. Grigorian, A. Tutela, and M. A. Zarbin, "Diabetic Macular Edema: Pathogenesis and Treatment," *Survey of Ophthalmology*, vol. 54, pp. 1–32, 1 2009, ISSN: 0039-6257. DOI: 10.1016/j.survophthal.2008.10.001.

[7] F. Bandello, R. Lattanzio, I. Zucchiatti, A. Arrigo, M. Battista, and M. V. Cicinelli, "Diabetic Macular Edema," in M. Attilio, L. Rosangela, Z. I. B. Francesco, and Zarbin, Eds. Springer International Publishing, 2019, pp. 97–183, ISBN: 978-3-319-96157-6. DOI: 10.1007/978-3-319-96157-6_3.

[8] T. Y. Wong and I. U. Scott, "Retinal-Vein Occlusion," *New England Journal of Medicine*, vol. 363, pp. 2135–2144, 22 2010. DOI: 10.1056/NEJMcp1003934.

[9] D. Huang et al., "Optical coherence tomography," *Science*, vol. 254, pp. 1178–1181, 5035 Nov. 1991. DOI: 10.1126/science.1957169.

[10] W. Drexler and J. G. Fujimoto, "State-of-the-art retinal optical coherence tomography," *Progress in Retinal and Eye Research*, vol. 27, pp. 45–88, 1 2008, ISSN: 1350-9462. DOI: 10.1016/j.preteyeres.2007.07.005.

[11] I. A. Viedma, D. Alonso-Caneiro, S. A. Read, and M. J. Collins, "Deep learning in retinal optical coherence tomography (OCT): A comprehensive survey," *Neurocomputing*, vol. 507, pp. 247–264, 2022, ISSN: 0925-2312. DOI: 10.1016/j.neucom.2022.08.021.

[12] N. Jain et al., "Quantitative Comparison of Drusen Segmented on SD-OCT versus Drusen Delineated on Color Fundus Photographs," *Investigative Ophthalmology & Visual Science*, vol. 51, pp. 4875–4883, 10 May 2010, ISSN: 1552-5783. DOI: 10.1167/iovs.09-4962.

[13] M. Almonte, P. Capellàn, T. Yap, and M. F. Cordeiro, "Retinal correlates of psychiatric disorders," *Therapeutic Advances in Chronic Disease*, vol. 11, p. 204 062 232 090 521, Dec. 2020. DOI: 10.1177/2040622320905215.

[14] E. López-Varela, N. Barreira, N. O. Pascual, M. R. A. Castillo, and M. G. Penedo, "Generation of synthetic intermediate slices in 3D OCT cubes for improving pathology detection and monitoring," *Computers in Biology and Medicine*, vol. 163, p. 107 214, 2023, ISSN: 0010-4825. DOI: 10.1016/j.compbiomed.2023.107214.

[15] E. Selvi, M. Özdemir, and M. A. Selver, "Performance Analysis of Distance Transform Based Inter-Slice Similarity Information on Segmentation of Medical Image Series," *Mathematical and Computational Applications*, vol. 18, pp. 511–520, 3 2013, ISSN: 2297-8747. DOI: 10.3390/mca18030511.

[16] T. C. Quek et al., "Predictive, preventive, and personalized management of retinal fluid via computer-aided detection app for optical coherence tomography scans," *EPMA Journal*, vol. 13, pp. 547–560, 4 2022, ISSN: 1878-5085. DOI: 10.1007/s13167-022-00301-5.

[17] S. J. Pawan et al., "Capsule Network–based architectures for the segmentation of sub-retinal serous fluid in optical coherence tomography images of central serous chorioretinopathy," *Medical & Biological Engineering & Computing*, vol. 59, pp. 1245–1259, 6 2021, ISSN: 1741-0444. DOI: 10.1007/s11517-021-02364-4.

[18] X. Liu, S. Wang, Y. Zhang, D. Liu, and W. Hu, "Automatic fluid segmentation in retinal optical coherence tomography images using attention based deep learning," *Neurocomputing*, vol. 452, pp. 576–591, 2021, ISSN: 0925-2312. DOI: 10.1016/j.neucom.2020.07.143.

[19] Y. Guo, T. T. Hormel, H. Xiong, J. Wang, T. S. Hwang, and Y. Jia, "Automated Segmentation of Retinal Fluid Volumes From Structural and Angiographic Optical Coherence Tomography Using Deep Learning," *Translational Vision Science & Technology*, vol. 9, p. 54, 2 Oct. 2020, ISSN: 2164-2591. DOI: 10.1167/tvst.9.2.54.

[20] Z. Wang et al., "Automated segmentation of macular edema for the diagnosis of ocular disease using deep learning method," *Scientific Reports*, vol. 11, p. 13 392, 1 2021, ISSN: 2045-2322. DOI: 10.1038/s41598-021-92458-8.

[21] Y. Wu et al., "Training Deep Learning Models to Work on Multiple Devices by Cross-Domain Learning with No Additional Annotations," *Ophthalmology*, vol. 130, pp. 213–222, 2 2023, ISSN: 0161-6420. DOI: 10.1016/j.ophtha.2022.09.014.

[22] M. Rahil, B. N. Anoop, G. N. Girish, A. R. Kothari, S. G. Koolagudi, and J. Rajan, "A Deep Ensemble Learning-Based CNN Architecture for Multiclass Retinal Fluid Segmentation in OCT Images," *IEEE Access*, vol. 11, pp. 17 241–17 251, 2023, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2023.3244922.

[23] H. Liu et al., "Semantic uncertainty Guided Cross-Transformer for enhanced macular edema segmentation in OCT images," *Computers in Biology and Medicine*, vol. 174, p. 108 458, 2024, ISSN: 0010-4825. DOI: 10.1016/j.compbiomed.2024.108458.

[24] X. Li, S. Niu, X. Gao, X. Zhou, J. Dong, and H. Zhao, "Self-training adversarial learning for cross-domain retinal OCT fluid segmentation," *Computers in Biology and Medicine*, vol. 155, p. 106 650, 2023, ISSN: 0010-4825. DOI: 10.1016/j.compbiomed.2023.106650.

[25] K. Gao et al., "Double-branched and area-constraint fully convolutional networks for automated serous retinal detachment segmentation in SD-OCT images," *Computer Methods and Programs in Biomedicine*, vol. 176, pp. 69–80, 2019, ISSN: 0169-2607. DOI: 10.1016/j.cmpb.2019.04.027.

[26] B. Hassan et al., "Deep learning based joint segmentation and characterization of multi-class retinal fluid lesions on OCT scans for clinical use in anti-VEGF therapy," *Computers in Biology and Medicine*, vol. 136, p. 104 727, 2021, ISSN: 0010-4825. DOI: 10.1016/j.compbiomed.2021.104727.

[27] D. Lu et al., "Deep-learning based multiclass retinal fluid segmentation and detection in optical coherence tomography images using a fully convolutional neural network," *Medical Image Analysis*, vol. 54, pp. 100–110, 2019, ISSN: 1361-8415. DOI: 10.1016/j.media.2019.02.011.

[28] B. Hassan, S. Qin, T. Hassan, R. Ahmed, and N. Werghi, "Joint Segmentation and Quantification of Chorioretinal Biomarkers in Optical Coherence Tomography Scans: A Deep Learning Approach," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–17, 2021. DOI: 10.1109/TIM.2021.3077988.

[29] H. Zhang, J. Yang, C. Zheng, S. Zhao, and A. Zhang, "Annotation-efficient learning for OCT segmentation," *Biomed. Opt. Express*, vol. 14, pp. 3294–3307, 7 Jul. 2023. DOI: 10.1364/BOE.486276.

[30] L. B. Sappa et al., "RetFluidNet: Retinal Fluid Segmentation for SD-OCT Images Using Convolutional Neural Network," *Journal of Digital Imaging*, vol. 34, pp. 691–704, 3 2021, ISSN: 1618-727X. DOI: 10.1007/s10278-021-00459-w.

[31] G. Xing et al., "Multi-Scale Pathological Fluid Segmentation in OCT With a Novel Curvature Loss in Convolutional Neural Network," *IEEE Transactions on Medical Imaging*, vol. 41, pp. 1547–1559, 6 2022. DOI: 10.1109/TMI.2022.3142048.

[32] W. Tang et al., "Multi-class retinal fluid joint segmentation based on cascaded convolutional neural networks," *Physics in Medicine & Biology*, vol. 67, p. 125 018, 12 Jun. 2022. DOI: 10.1088/1361-6560/ac7378.

[33] F. D. Padilla-Pantoja, Y. D. Sanchez, B. A. Quijano-Nieto, O. J. Perdomo, and F. A. Gonzalez, "Etiology of Macular Edema Defined by Deep Learning in Optical Coherence Tomography Scans," *Translational Vision Science & Technology*, vol. 11, p. 29, 9 Sep. 2022, ISSN: 2164-2591. DOI: 10.1167/tvst.11.9.29.

[34] J. Hu, Y. Chen, and Z. Yi, "Automated segmentation of macular edema in OCT using deep neural networks," *Medical Image Analysis*, vol. 55, pp. 216–227, 2019, ISSN: 1361-8415. DOI: 10.1016/j.media.2019.05.002.

[35] I. Mantel et al., "Automated Quantification of Pathological Fluids in Neovascular Age-Related Macular Degeneration, and Its Repeatability Using Deep Learning," *Translational Vision Science & Technology*, vol. 10, p. 17, 4 Apr. 2021, ISSN: 2164-2591. DOI: 10.1167/tvst.10.4.17.

[36] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds., Springer International Publishing, 2015, pp. 234–241, ISBN: 978-3-319-24574-4. DOI: 10.1007/978-3-319-24574-4_28.

[37] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, pp. 834–848, 4 2018. DOI: 10.1109/TPAMI.2017.2699184.

[38] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv e-prints*, arXiv:1409.1556, Sep. 2014. DOI: 10.48550/arXiv.1409.1556.

[39] C. You et al., "CT Super-Resolution GAN Constrained by the Identical, Residual, and Cycle Learning Ensemble (GAN-CIRCLE)," *IEEE Transactions on Medical Imaging*, vol. 39, pp. 188–203, 1 2020. DOI: 10.1109/TMI.2019.2922960.

[40] M. Ibrahim et al., "Generative AI for Synthetic Data Across Multiple Medical Modalities: A Systematic Review of Recent Developments and Challenges," *arXiv e-prints*, arXiv:2407.00116, Jun. 2024. DOI: 10.48550/arXiv.2407.00116.

[41] Y. Xia, N. Ravikumar, J. P. Greenwood, S. Neubauer, S. E. Petersen, and A. F. Frangi, "Super-Resolution of Cardiac MR Cine Imaging using Conditional GANs and Unsupervised Transfer Learning," *Medical Image Analysis*, vol. 71, p. 102 037, 2021, ISSN: 1361-8415. DOI: 10.1016/j.media.2021.102037.

[42] Z. Wu, J. Wei, J. Wang, and R. Li, "Slice imputation: Multiple intermediate slices interpolation for anisotropic 3D medical image segmentation," *Computers in Biology and Medicine*, vol. 147, p. 105 667, 2022, ISSN: 0010-4825. DOI: 10.1016/j.compbiomed.2022.105667.

[43] S. Nishimoto, K. Kawai, K. Nakajima, H. Ishise, and M. Kakibuchi, "Generating intermediate slices with U-nets in craniofacial CT images," *medRxiv*, p. 2024.05.08.24307089, Jan. 2024. DOI: 10.1101/2024.05.08.24307089.

[44] H. Zhang, X. Yang, Y. Cui, Q. Wang, J. Zhao, and D. Li, "A novel GAN-based three-axis mutually supervised super-resolution reconstruction method for rectal cancer MR image," *Computer Methods and Programs in Biomedicine*, vol. 257, p. 108 426, 2024, ISSN: 0169-2607. DOI: 10.1016/j.cmpb.2024.108426.

[45] C. Peng, W.-A. Lin, H. Liao, R. Chellappa, and S. K. Zhou, "SAINT: Spatially Aware Interpolation NeTwork for Medical Slice Synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020. DOI: 10.48550/arXiv.2001.00704.

[46] C. Fang, L. Wang, D. Zhang, J. Xu, Y. Yuan, and J. Han, "Incremental Cross-View Mutual Distillation for Self-Supervised Medical CT Synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2022, pp. 20 677–20 686. DOI: 10.48550/arXiv.2112.10325.

[47] U. Nimitha and P. Ameer, "MRI super-resolution using similarity distance and multi-scale receptive field based feature fusion GAN and pre-trained slice interpolation network," *Magnetic Resonance Imaging*, vol. 110, pp. 195–209, 2024, ISSN: 0730-725X. DOI: 10.1016/j.mri.2024.04.021.

[48] M.-I. Georgescu, R. T. Ionescu, and N. Verga, "Convolutional Neural Networks With Intermediate Loss for 3D Super-Resolution of CT and MRI Scans," *IEEE Access*, vol. 8, pp. 49 112–49 124, 2020. DOI: 10.1109/ACCESS.2020.2980266.

[49] Y. Chen, F. Shi, A. G. Christodoulou, Y. Xie, Z. Zhou, and D. Li, "Efficient and Accurate MRI Super-Resolution Using a Generative Adversarial Network and 3D Multi-level Densely Connected Network," in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*, A. F. Frangi, S. Julia A., C. Davatzikos, C. Alberola-López, and G. Fichtinger, Eds., Springer International Publishing, 2018, pp. 91–99, ISBN: 978-3-030-00928-1. DOI: 10.1007/978-3-030-00928-1_11.

[50] I. Sanchez and V. Vilaplana, "Brain MRI super-resolution using 3D generative adversarial networks," *arXiv e-prints*, arXiv:1812.11440, Dec. 2018. DOI: 10.48550/arXiv.1812.11440.

[51] A. Kudo, Y. Kitamura, Y. Li, S. Iizuka, and E. Simo-Serra, "Virtual Thin Slice: 3D Conditional GAN-based Super-Resolution for CT Slice Interval," in *Machine Learning for Medical Image Reconstruction*, Andreas, R. Daniel, Y. J. C. K. Florian, and Maier, Eds., Springer International Publishing, 2019, pp. 91–100, ISBN: 978-3-030-33843-5. DOI: 10.1007/978-3-030-33843-5_9.

[52] K. Zhang et al., "SOUP-GAN: Super-Resolution MRI Using Generative Adversarial Networks," *Tomography*, vol. 8, pp. 905–919, 2 2022, ISSN: 2379-139X. DOI: 10.3390/tomography8020073.

[53] B. Niu et al., "Single Image Super-Resolution via a Holistic Attention Network," in *Computer Vision – ECCV 2020*, Horst, B. Thomas, F. J.-M. V. Andrea, and Bischof, Eds., Springer International Publishing, 2020, pp. 191–207, ISBN: 978-3-030-58610-2. DOI: 10.1007/978-3-030-58610-2_12.

[54] T. Ding, L. Liang, Z. Zhu, and I. Zharkov, "CDFI: Compression-Driven Network Design for Frame Interpolation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2021, pp. 8001–8011. DOI: 10.48550/arXiv.2103.10559.

[55] H. Lee, T. Kim, T.-y. Chung, D. Pak, Y. Ban, and S. Lee, "AdaCoF: Adaptive Collaboration of Flows for Video Frame Interpolation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020. DOI: 10.48550/arXiv.1907.10244.

[56] L. Gambini et al., "Video frame interpolation neural network for 3D tomography across different length scales," *Nature Communications*, vol. 15, p. 7962, 1 2024, ISSN: 2041-1723. DOI: 10.1038/s41467-024-52260-2.

[57] Z. Huang, T. Zhang, W. Heng, B. Shi, and S. Zhou, "Real-Time Intermediate Flow Estimation for Video Frame Interpolation," in *Computer Vision – ECCV 2022*, S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Eds., Springer Nature Switzerland, 2022, pp. 624–642, ISBN: 978-3-031-19781-9. DOI: 10.1007/978-3-031-19781-9_36.

[58] Q. N. Tran and S.-H. Yang, "Efficient Video Frame Interpolation Using Generative Adversarial Networks," *Applied Sciences*, vol. 10, 18 2020, ISSN: 2076-3417. DOI: 10.3390/app10186245.

[59] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-To-Image Translation With Conditional Adversarial Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017. DOI: 10.48550/arXiv.1611.07004.

[60] H. Bogunović et al., "RETOUCH: The Retinal OCT Fluid Detection and Segmentation Benchmark and Challenge," *IEEE Transactions on Medical Imaging*, vol. 38, pp. 1858–1874, 8 2019. DOI: 10.1109/TMI.2019.2901398.

[61] S. J. Chiu, M. J. Allingham, P. S. Mettu, S. W. Cousins, J. A. Izatt, and S. Farsiu, "Kernel regression based segmentation of optical coherence tomography images with diabetic macular edema," *Biomed. Opt. Express*, vol. 6, pp. 1172–1194, 4 Apr. 2015. DOI: 10.1364/BOE.6.001172.

[62] A. Rashno et al., "Fully-automated segmentation of fluid regions in exudative age-related macular degeneration subjects: Kernel graph cut in neutrosophic domain," *PLOS ONE*, vol. 12, pp. 1–26, 10 Dec. 2017. DOI: 10.1371/journal.pone.0186949.

[63] A. Rashno et al., "Fully Automated Segmentation of Fluid/Cyst Regions in Optical Coherence Tomography Images With Diabetic Macular Edema Using Neutrosophic Sets and Graph Algorithms," *IEEE Transactions on Biomedical Engineering*, vol. 65, pp. 989–1001, 5 2018. DOI: 10.1109/TBME.2017.2734058.

[64] R. R. Shamir, Y. Duchin, J. Kim, G. Sapiro, and N. Harel, "Continuous Dice Coefficient: a Method for Evaluating Probabilistic Segmentations," *arXiv e-prints*, arXiv:1906.11031, Jun. 2019. DOI: 10.48550/arXiv.1906.11031.

[65] R. Tennakoon, A. K. Gostar, R. Hoseinnezhad, and A. Bab-Hadiashar, "Retinal fluid segmentation in OCT images using adversarial loss based convolutional neural networks," in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, 2018, pp. 1436–1440. DOI: 10.1109/ISBI.2018.8363842.

[66] D. P. Kingma, "Adam: A method for stochastic optimization," in *3rd International Conference for Learning Representations*, 2015. DOI: 10.48550/arXiv.1412.6980.

[67] U. Sara, M. Akter, and M. S. Uddin, "Image quality assessment through FSIM, SSIM, MSE and PSNR—a comparative study," *Journal of Computer and Communications*, vol. 7, pp. 8–18, 3 2019. DOI: 10.4236/jcc.2019.73002.

[68] S. Rajkumar and G. Malathi, "A comparative analysis on image quality assessment for real time satellite images," *Indian J. Sci. Technol*, vol. 9, pp. 1–11, 34 2016. DOI: 10.17485/ijst/2016/v9i34/96766.

[69] I. Goodfellow et al., "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Eds., vol. 27, Curran Associates, Inc., 2014. DOI: 10.48550/arXiv.1406.2661.

[70] I. Goodfellow et al., "Generative adversarial networks," *Commun. ACM*, vol. 63, pp. 139–144, 11 Oct. 2020, ISSN: 0001-0782. DOI: 10.1145/3422622.

[71] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative Adversarial Networks: An Overview," *IEEE Signal Processing Magazine*, vol. 35, pp. 53–65, 1 2018. DOI: 10.1109/MSP.2017.2765202.

[72] Z. Wang, E. Simoncelli, and A. Bovik, "Multiscale structural similarity for image quality assessment," in *Conference Record of the Asilomar Conference on Signals, Systems and Computers*, vol. 2, May 2003, 1398–1402 Vol.2, ISBN: 0-7803-8104-1. DOI: 10.1109/ACSSC.2003.1292216.

[73] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," in *4th International Conference on Learning Representations, ICLR 2016*, 2016. DOI: 10.48550/arXiv.1511.05440.

[74] J. Åkesson, J. Töger, and E. Heiberg, "Random effects during training: Implications for deep learning-based medical image segmentation," *Computers in Biology and Medicine*, vol. 180, p. 108 944, 2024, ISSN: 0010-4825. DOI: doi.org/10.1016/j.compbiomed.2024.108944.

[75] M. G. Altarabichi, S. Nowaczyk, S. Pashami, P. S. Mashhadi, and J. Handl, "Rolling the dice for better deep learning performance: A study of randomness techniques in deep neural networks," *Information Sciences*, vol. 667, p. 120 500, 2024, ISSN: 0020-0255. DOI: 10.1016/j.ins.2024.120500.

Appendix A

Variability of the Segmentation U-Net

Table A.1: Dice scores for every vendor and fluid for multiple runs performed in the same conditions. "Set 1" resumes Run 1 to Run 5, all of which validated using fold 2. Meanwhile, "Set 2" resumes the Runs from 6 to 14, validated in fold 3. All the runs in each set were trained on the same conditions, for 100 epochs. The random transformations applied were horizontal flipping and maximum rotation of 10° applied to it. This provides an insight of how the randomness of the U-Net affects the segmentation performance.

Runs	**VF**	**Cirrus**			**Spectralis**			**Topcon**			**IRF**	**SRF**	**PED**	**Fluid**
		IRF	**SRF**	**PED**	**IRF**	**SRF**	**PED**	**IRF**	**SRF**	**PED**				
Run 1	2	0.555	0.773	0.634	0.849	0.857	0.839	0.783	0.869	0.765	0.686	0.821	0.716	0.663
Run 2	2	0.490	0.813	0.698	0.786	0.848	0.856	0.772	0.903	0.834	0.639	0.850	0.773	0.692
Run 3	2	0.349	0.774	0.600	0.544	0.799	0.759	0.674	0.894	0.772	0.494	0.819	0.687	0.561
Run 4	2	0.542	0.776	0.677	0.831	0.826	0.822	0.852	0.873	0.856	0.699	0.818	0.764	0.685
Run 5	2	0.611	0.817	0.626	0.832	0.836	0.834	0.850	0.913	0.822	0.732	0.853	0.730	0.685
Set 1	2	0.51	0.79	0.65	0.77	0.83	0.82	0.79	0.89	0.81	0.65	0.83	0.73	0.66
Run 6	3	0.661	0.746	0.495	0.632	0.842	0.736	0.702	0.875	0.728	0.671	0.810	0.622	0.617
Run 7	3	0.688	0.622	0.596	0.665	0.707	0.728	0.846	0.851	0.747	0.742	0.721	0.675	0.634
Run 8	3	0.794	0.862	0.632	0.663	0.887	0.778	0.874	0.833	0.780	0.801	0.856	0.712	0.719
Run 9	3	0.622	0.822	0.682	0.677	0.846	0.758	0.736	0.853	0.741	0.673	0.838	0.717	0.685
Run 10	3	0.649	0.718	0.671	0.653	0.872	0.760	0.828	0.824	0.792	0.715	0.784	0.731	0.700
Run 11	3	0.607	0.802	0.614	0.669	0.841	0.760	0.837	0.864	0.787	0.702	0.831	0.703	0.697
Run 12	3	0.565	0.788	0.671	0.550	0.868	0.755	0.784	0.831	0.773	0.643	0.818	0.723	0.648
Run 13	3	0.618	0.699	0.406	0.471	0.724	0.630	0.752	0.878	0.645	0.641	0.769	0.533	0.531
Run 14	3	0.757	0.815	0.701	0.646	0.894	0.773	0.881	0.848	0.836	0.783	0.841	0.763	0.738
Set 2	3	0.66	0.76	0.61	0.63	0.83	0.74	0.80	0.85	0.76	0.71	0.81	0.69	0.66