UNIVERSIDADE D
COIMBRA

Joana Margarida Moreira Pereira de Moura

Automatic Segmentation Of Retinal Layers In Optical Coherence Tomography Images:
A Flexible Model For Multiple Animal Models And Acquisition Systems

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering Physics, supervised by Professor Doctor Rui Bernardes and Doctor Pedro Guimarães, to the Department of Physics of the Faculty of Sciences and Technology of the University of Coimbra.

September, 2023

Faculty of Sciences and Technology of the University of Coimbra
Department of Physics

AUTOMATIC SEGMENTATION OF RETINAL LAYERS IN OPTICAL COHERENCE TOMOGRAPHY IMAGES:

A FLEXIBLE MODEL FOR MULTIPLE ANIMAL MODELS AND ACQUISITION SYSTEMS

Joana Margarida Moreira Pereira de Moura

A thesis presented to obtain the Master's degree in Engineering Physics, supervised by Professor Doctor Rui Bernardes and Doctor Pedro Guimarães, to the Department of Physics of the Faculty of Sciences and Technology of the University of Coimbra.

September, 2023

1 2 9 0

UNIVERSIDADE D
COIMBRA

Agradecimentos

Gostaria de começar por agradecer aos meus orientadores, o Doutor Pedro Guimarães e o Professor Doutor Rui Bernardes, pelo seu apoio ao longo do desenvolvimento desta dissertação. Agradeço toda a disponibilidade e orientação fornecidas, o conhecimento transmitido e a colaboração no solucionar de dúvidas e problemas que foram surgindo ao longo da realização deste trabalho. Sem o seu acompanhamento, esta tese não seria possível.

Aos meus pais agradeço profundamente a dedicação, educação e orgulho nas minhas conquistas. Acima de tudo, obrigado por terem contribuído para o meu sucesso e para o alcance desta etapa.

Queria também agradecer aos meus amigos, por serem as pessoas fantásticas que são e por terem passado comigo estes cinco anos de muita felicidade e amizade. Levarei sempre comigo todas as memórias que passámos, os jantares, as saídas à noite na alta, as sessões de estudo na BG, as viagens... A todas as nossas aventuras, um grande obrigado.

Por fim, um obrigado especial ao António, por toda a paciência, compreensão e apoio nos momentos mais difíceis, e por ter estado sempre a meu lado neste longo percurso.

Abstract

The retina can be used as a window to the brain and is accessible through optical methods. As such, the analysis of changes detected in retinal images allows for the non-invasive diagnosis of neurodegenerative diseases, which remains complex and challenging. Advanced medical image analysis processes, such as deep learning (DL), are growing rapidly and have enabled the development of various retinal segmentation methods. However, these solutions are created based on a specific type of data and/or acquisition system. The creation of a flexible and versatile segmentation model that is capable of segmenting images of retinas from different disease models and acquisition systems is important, especially for applications such as the individual analysis of retinal layers and the measurement of their thickness, which help in the diagnosis and monitoring of various diseases.

The aim of the work proposed here is to segment optical coherence tomography (OCT) images for application in the retinas of multiple animal models of neurodegenerative diseases and multiple acquisition systems, using DL algorithms.

For this purpose, two convolution neural networks were created, the Attention-Res-U-Net and the Teacher-Student Generative Adversarial Network (GAN), which were trained on a set of B-scans obtained in mice, controls and models of Alzheimer's disease, with the Phoenix OCT imaging system.

In the test set, with the same types of controls and disease models, a mean absolute error (MAE) of 0.88 μm with an interquartile range (IQR) of 0.42 μm was achieved for the Attention-Res-U-Net and an MAE of 0.54 μm with an IQR of 0.29 μm for the Teacher-Student GAN, these values being the distances between the automatic segmentation and

the previously existing segmentations. Due to resource management, the remaining tests were only carried out with the Attention-Res-U-Net. Thus, another test was carried out, this time with a dataset of animal models of diabetes, where the network achieved an MAE of 5.09 μm with an IQR of 4.38 μm. In the last phase, transfer learning was applied to the trained Attention-Res-U-Net model, with the goal of applying it to systems other than the one used for training, in this case, the Bioptigen. This new model was also tested with B-scans obtained from the new system, and successfully segmented the intended layers, according to expert evaluation.

In general, the developed segmentation algorithm was able to successfully segment the retinal layers in the images of the different animal models, in both acquisition systems.

Keywords: Machine Learning, Optical Coherence Tomography, Segmentation, Retina, Central Nervous System.

Resumo

A retina pode ser utilizada como uma janela para o cérebro e é acessível através de métodos óticos. Como tal, a análise de alterações detetadas em imagens da retina permite o diagnóstico não invasivo de doenças neurodegenerativas, o qual continua a ser complexo e desafiante. Processos avançados de análise de imagens médicas, como o *deep learning* (DL), estão em franco crescimento e têm permitido o desenvolvimento de diversos métodos de segmentação da retina. No entanto, estas soluções são criadas com base num tipo específico de dados e/ou sistema de aquisição. A criação de um modelo de segmentação flexível e versátil que seja capaz de segmentar imagens de retinas de diferentes modelos de doença e sistemas de aquisição, é importante, especialmente para aplicações como a análise individual das camadas da retina e a medição da sua espessura, que ajudam no diagnóstico e monitorizaçáo de diversas doenças.

O trabalho aqui proposto tem como finalidade a segmentação de imagens de tomografia de coerência ótica (OCT) para aplicação em retinas de múltiplos modelos animais de doenças neurodegenerativa e múltiplos sistemas de aquisição, através algoritmos DL.

Para tal, foram criadas duas redes neuronais de convolução, a *Attention-Res-U-Net* e a *Teacher-Student Generative Adversarial Network*, as quais foram treinadas num conjunto de *B-scans* obtidos em murganhos, controlos e modelos da doença de *Alzheimer*, com o sistema de imagem *Phoenix OCT*.

No conjunto de teste, com os mesmos tipos de controlos e modelos de doença, foi atingido um *mean absolute error* (MAE) de 0.88 μm com uma interquartile range (IQR) de 0.42 μm para a *Attention-Res-U-Net* e um MAE de 0.54 μm com uma IQR de 0.29 μm para a *Teacher-Student GAN*, sendo estes valores as distâncias entre a segmentação

automática e as segmentações previamente existentes. Devido a uma gestão de recursos, os restantes testes foram realizados apenas com a *Attention-Res-U-Net*. Assim, outro teste foi realizado, desta vez com um dataset de modelo animal diabético, onde a rede atingiu um MAE de 5.09 μm com uma IQR de 4.38 μm. Numa última fase, foi aplicado *transfer learning* ao modelo *Attention-Res-U-Net* treinado, com vista à sua aplicação a sistemas distintos do usado no treino, neste caso o *Bioptigen*. Este novo modelo foi também testado com *B-scans* obtidos no novo sistema, tendo segmentado com sucesso as camadas pretendidas, de acordo com avaliação de experts.

Em geral, o algoritmo de segmentação desenvolvido foi capaz de segmentar com sucesso as camadas da retina nas imagens dos diferentes modelos animais, em ambos os sistemas de aquisição.

Palavras-chave: *Machine Learning*, Tomografia de Coerência Ótica, Segmentação, Retina, Sistema Nervoso Central.

Contents

Acronyms

Adagrad Adaptive Gradient Algorithm.

AI Artificial Intelligence.

AMD Age-related Macular Degeneration.

ANNs Artificial Neural Networks.

BCE Binary Cross-Entropy Loss.

BM Bruch's Membrane.

CIBIT Coimbra Institute for Biomedical Imaging and Translational Research.

CNN Convolutional Neural Network.

CNS Central Nervous System.

FC Fully-Connected.

FCM Fuzzy C-Means.

GAN Generative Adversarial Network.

GD Gradient Descent.

iCBR Coimbra Institute for Clinical and Biomedical Research.

ICNAS Institute of Applied Nuclear Sciences for Health.

IDE Integrated Development Environment.

INL Inner Nuclear Layer.

IoU Intersect over Union.

IPL Inner Plexiform Layer.

IS Inner Segments.

ISE Inner Segment Ellipsoid.

KS Kolmogorov-Smirnov.

MAE Mean Absolute Error.

MLP Multi-Layer Perceptron.

MSE Mean Squared Error.

NN Neural Network.

OCT Optical Coherence Tomography.

OD Right Eyes.

ONL Outer Nuclear Layer.

OPL Outer Plexiform Layer.

OPLD Optical path length difference.

OS Outer Segments.

ReLU Rectified Linear Unit.

RMSProp Root Mean Square Propagation.

RNFL-GCL Retinal Nerve Fiber Layer-Ganglion Cell.

ROI Region of Interest.

RPE Retinal Pigment Epithelium.

RPEDC RPE and Drusen Complex.

SD-OCT Spectral Domain OCT.

SGD Stochastic Gradient Descent.

SLP Single Layer Perceptron.

SVM Support Vector Machine.

WCE Weighted Categorical Cross-Entropy.

WT Wild-Type.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context and Motivation

Acquiring images of the central nervous system (CNS) has allowed us to expand our knowledge regarding the changes that occur in the CNS, not only in disease but also in healthy aging.

The retina, being part of the central nervous system, can be used as a window to the brain. The analysis of changes detected in images of the retina provides information about the central nervous system [1]. Currently, retinal imaging allows obtaining optical images through non-invasive methods of the CNS, which is not the case for the brain [2].

Therefore, one of the potential applications of examining the retina to spot changes in the CNS is the non-invasive diagnosis of neurodegenerative diseases.

Nowadays, the diagnosis of neurodegenerative diseases such as Alzheimer's disease, remains complex and challenging. In fact, diagnosing CNS diseases using traditional methods only detects symptoms when the condition is already in an advanced stage.

According to the World Health Organization (WHO), more than 55 million people are currently living with dementia, with approximately 10 million new cases every year. The most common form of dementia is Alzheimer's disease, accounting for 60-70% of cases [3].

As a result, it is crucial to seek early diagnosis methods that enable the identification of neuropathologies at the onset of the disease and the tracking of its progression. This may allow interventional measures to be taken in the time window when intervention could have a significant impact on delaying or even stopping the progression of the disease.

Several neurodegenerative diseases affect the retina. Therefore, examining the retina as a model for studying the central nervous system constitutes an excellent non-invasive alternative diagnostic method, as several studies have already demonstrated [4, 5]. This approach can potentially enhance our understanding of the CNS and the treatment of diseases by expanding the diagnosis and reaching a larger population.

One of the imaging techniques that have been increasingly used is the Optical Coherence Tomography (OCT). OCT is a crucial tool in clinical diagnosis as it enables the detailed visualization of the retina layered structure [2].

The OCT technique involves low-coherence light backscattering. It allows the acquisition of high-resolution retinal structural representations and easy access to detailed information about the microstructure of the retina. Additionally, it is extremely sensitive to any variations in the content and organization of its structures [6].

Recently, the Coimbra Institute for Biomedical Imaging and Translational Research (CIBIT) research center acquired a new imaging system, the Bioptigen OCT [7], with enhanced features, better resolution, and higher sampling per unit area than the existing Phoenix OCT system [8].

The analysis of medical images has been playing an increasingly important role in the healthcare sector and is on the rise [9]. Recent advances in medical image segmentation have enabled the development of new intelligent diagnostics. Machine learning, specifically deep learning (neural networks) has been applied to developing artificial intelligence (AI)-based diagnostic tools for medical images because it has demonstrated the ability to achieve high-accuracy diagnostic results [10, 11].

However, much of the research in this field has primarily concentrated on tackling segmentation within particular datasets, each associated with a single acquisition system. Consequently, it is essential to investigate segmentation techniques capable of delivering consistent performance across various disease models and acquisition systems.

The work herein was conducted to successfully segment OCT images from the retina of various animal models of disease and various imaging acquisition systems with different characteristics. In this context, it was crucial to optimize the segmentation process but, more importantly, to ensure that the system could be applied to a variety of disease models and different image acquisition systems. This issue is still open, to the best of our knowledge because, although there are image segmentation solutions for established data groups (images/animals), the problem under study has not yet been addressed and remains unresolved.

The work was carried out at the Institute of Applied Nuclear Sciences for Health (ICNAS) facilities. The project was overseen by Professor Doctor Rui Bernardes (CIBIT, ICNAS, FMUC) and Doctor Pedro Guimarães (CIBIT, ICNAS, UC).

1.2 Objectives

The project focuses on the segmentation of different layers of the neuroretina in B-scan images obtained by OCT.

With this objective in mind, the intention was to create and train a convolutional neural network whose output was segmenting the intended neuroretinal layers from B-scan

images collected from control mice (wild-type mice) and mice models of diseases, such as Alzheimer's and diabetes.

Data acquisition was carried out at the Coimbra Institute for Clinical and Biomedical Research (iCBR) and the Coimbra Institute for Biomedical Imaging and Translational Research (CIBIT) using two different imaging acquisition systems, respectively the Phoenix OCT and the Bioptigen. The collected data contains information from three groups of rats and mice: neurodegenerative disease models of Alzheimer's disease (3×Tg-AD), diabetes (type 1), and wild-type (WT) animals.

To overcome the limitations of the database, such as the lack of data variability and representativeness, processes for artificially increasing the quantity and diversity of data (data augmentation) were implemented. By introducing new variations of the existing images, the model will be forced to generalize, mitigating potential overfitting.

Subsequently, segmentation was achieved by implementing a convolutional neural network (CNN), whose architecture was also a subject of study. All neural networks were built, trained, and evaluated in Python.

Ensuring the correct segmentation of B-scan images and a solid ability to discriminate between the interfaces of the various layers is a crucial factor, as the primary purpose is the individual analysis of each neuroretinal layer.

Simultaneously, the knowledge gained after solving the proposed problem, i.e., the constructed neural network, should ideally allow for its successful application to future datasets. It will be of greater importance that the algorithm's architecture is flexible and capable of segmenting images from different instruments, various animal models of disease, and wild-type animals.

1.3 Requisite Analysis

1.3.1 Functional Requisites

The built system will have as its main function the segmentation of the following layers of the retina in OCT B-scan images of rats and mice using convolutional neural networks:

- RNFL and GCL (RNF-GCL): Retinal Nerve Fiber Layer and Ganglion Cell Layer[1];
- IPL: inner plexiform layer;
- INL: inner nuclear layer;
- OPL: outer plexiform layer;
- ONL: outer nuclear layer;
- ILS: photoreceptor inner segments;
- OLS: photoreceptor outer segments;
- RPE: retinal pigment epithelium.

1.3.2 Non-Functional Requisites

In terms of non-functional requirements, the following specifications have been defined to consider in the design and implementation of the system to be developed:

- Average error for any interface less than 2 micrometers per B-scan;
- Maximum error in any A-scan interface less than 10 micrometers;
- Applicable to wild-type (WT) rats and mice;

[1]The discrimination between the two is not possible for all animal groups considered in the present study

- Applicable to mice models of Alzheimer's disease (type 3×Tg-AD);
- Applicable to rats models of diabetes (type 1);
- Applicable to images obtained for ages ranging from 1 to 4 months;
- Development based on data obtained from the Phoenix OCT imaging system;
- Application of transfer learning methods for use on data obtained from the Bioptigen imaging system.

1.4 Document Outline

The document's structure is divided into six chapters. The organization of information in the remaining chapters will follow the following structure:

- Chapter 1 established the context, motivation, and primary objectives as well as the requisite analysis and document outline;
- Chapter 2 will delve into concepts related to retinal imaging and provide a brief overview of the structure of the retina. Furthermore, it will delve into the theory of neural networks, with a particular focus on convolutional neural networks;
- In Chapter 3 various methods and techniques that are applied to the problem of segmentation of retinal layers are going to be briefly analyzed;
- In Chapter 4 the development of the algorithm will be explained in detail, as well as all the methods used and stages that it underwent;
- Chapter 5 will present the achieved results and their discussion.
- Finally, Chapter 6 will present conclusions and recommendations for future work.

Chapter 2

Fundamental Concepts

2.1 Retina

The retina is a thin, delicate tissue sensitive to light, located at the back of the eye. This tissue structure is part of the central nervous system and contains millions of photosensitive cells that respond to the light, capturing and converting it into electrical signals that are transmitted to the brain through the fibers of the optic nerve. These signals are processed and interpreted in the brain as visual images, making the retina essential to human vision as it plays a pivotal role in the visual process [12].

Visually, the retina has a layered structure, as shown in Figure 2.1. Listed from the innermost layer to the outermost layer, the main layers are [13]:

- **Retinal nerve fiber layer (RNFL or NFL)**: This layer comprises the bundled axons of ganglion cells, forming the optic nerve;
- **Ganglion cell layer (GCL)**: The ganglion cell layer is composed exclusively of ganglion cells, which are the output neurons of the retina;
- **Inner plexiform layer (IPL)**: In the inner plexiform layer, bipolar cells transmit information from cones and rods to ganglion cells;

- **Inner nuclear layer (INL)**: The inner nuclear layer contains cells that provide feedback regulation for cones and rods;
- **Outer plexiform layer (OPL)**: In the outer plexiform layer, photoreceptor cells relay information to the extensions of nerve cells;
- **Outer nuclear layer (ONL)**: The outer nuclear layer houses the cell bodies of rods and cones;
- **Photoreceptor layer:** The photoreceptor layer encompasses the **inner segments** (**ILS** or PIS) and **outer segments** (**OLS** or POS) responsible for converting light into electrical signals;
- **Retinal pigment epithelium (RPE)**: The retinal pigment epithelium, located at the outermost layer of the retina, between the choroid and the photoreceptor cells, provides nourishment to the photoreceptor cells and aids in waste removal.

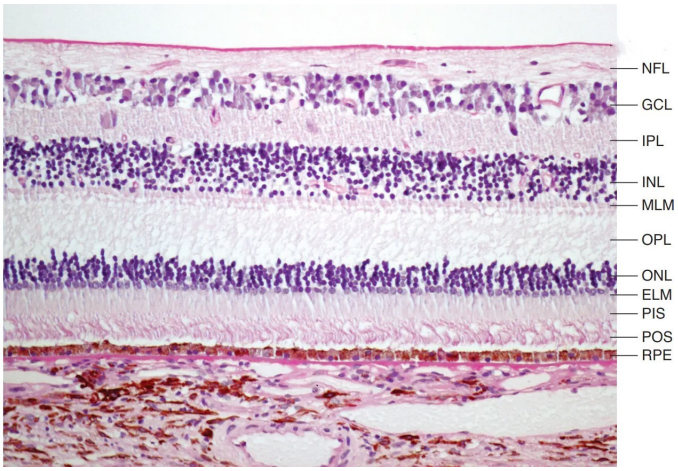


Figure 2.1: Histological cross-section of the human retina [13]. The basic retinal structure is arranged in different layers, from retinal nerve fiber layer (NFL), ganglion cell layer (GCL), inner plexiform layer (IPL), inner nuclear layer (INL), outer plexiform layer (OPL), outer nuclear layer (ONL), external limiting membrane (ELM), photoreceptor inner segments (PIS), photoreceptor outer segments (POS), and retinal pigment epithelium (RPE).

Various diseases and conditions can affect the retina, rendering its structure susceptible to alterations. As previously indicated, the evaluation of the retina can serve as a valuable diagnostic tool for a range of pathologies. Healthcare professionals often use techniques

such as retinal imaging and examinations to gather critical insights into a patient's overall health, through the detailed visualization of the retina's structure [2].

2.2 Retinal Imaging: Optical Coherence Tomography

Retinal imaging techniques encompass a wide range of advanced methods that enable the visualization and analysis of the retina's intricate structures and functions. In the field of ophthalmology, these techniques play a crucial role in diagnosing various conditions, monitoring disease progression, and guiding treatment decisions [14, 5, 4].

Several types of imaging technologies allow the observation and capture of images of the retina. Such methods include optical coherence tomography (OCT), OCT angiography, fundus photography, fluorescein angiography, indocyanine green angiography, retinal scanning laser polarimetry, and more. This study specifically focuses on OCT.

Optical Coherence Tomography (OCT)

Optical coherence tomography [6] is a non-invasive imaging exam of biological tissues, allowing the in-vivo collection of cross-sectional images of the tissues, most notably the eye's retinal layers. This technique enables the detection and analysis of subtle changes in retinal thickness, morphology, and fluid accumulation before symptoms become clinically apparent.

OCT is based on the principle of low-coherence interferometry [15]. It involves emitting a beam of low-coherence near-infrared light that is split into a sample beam, which interacts with the tissue of interest, and a reference beam, which follows a known path length. When the sample beam interacts with the tissue, it is partially backscattered. The backscattered light waves travel from different optical path lengths, depending on the depth within the tissue from which they are backscattered. The reference beam interferes with the reflected light waves, where they are combined, creating interference patterns that are measured by a detector.

By analyzing these interference patterns, OCT can provide depth information about the eye's internal structures, if the studied tissue is the eye. When two light beams are combined, they add constructively or destructively. For high-bandwidth, low-coherence light, constructive interference occurs only when the difference between the optical paths is less than the coherence of the light. By changing the length of the reference beam OCT can scan the sample (A-scan).

In the Spectral Domain OCT (SD-OCT) [16], which is the most commonly used OCT technique, a spectrometer is used in place of the detector, to disperse the interference spectrum (combined light) into its constituent wavelengths and the intensity of light at a specific wavelength is detected, which is then analyzed in the frequency domain. The main advantage is that the reference mirror is fixed. Since there are no moving parts acquisition time is much faster. This process involves applying a Fourier transformation to the measured spectral interference data, converting the spectral information from the frequency domain into the depth or spatial domain. The position of spectral peaks in the resulting interference spectrum corresponds to a different optical path length (depth) within the sample. With this, depth-resolved information about the sample is extracted, representing the reflectivity or scattering intensity of light as a function of the sample's depth [17]. This depth-resolved information is used to construct cross-sectional images (B-scans) of the sample. Each of the depth profiles obtained (A-scans) corresponds to a single column in a cross-sectional image (B-scan).

Each B-scan represents a single lateral slice of the tissue, showing the internal structures and layers at different depths. To create a 3D representation of the tissue, multiple B-scans are acquired rapidly in succession at adjacent lateral positions, allowing for the construction of volumetric images (3D OCT) that provide a more comprehensive view of the tissue.

With the evolution of SD-OCT systems [18], these devices can image the retina at a high speed, with reported axial resolutions of 4 to 7μm [19, 20, 21], enabling the visualization of intricate retinal structures. It is one of the most popular supplementary

exams used in the diagnosis of retinal pathologies, not only for its high resolution and speed but also for its simplicity and non-invasiveness [11]. Figure 2.2 represents a B-scan with segmented retinal layers.

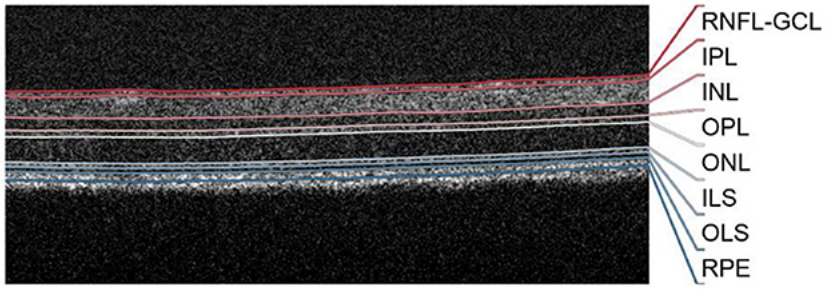


Figure 2.2: Retinal layers' interfaces on a B-scan of a mouse retina [22]. From top to bottom: retinal nerve fiber layer - ganglion cell layer (RNFL-GCL), inner plexiform layer (IPL), inner nuclear layer (INL), outer plexiform layer (OPL), outer nuclear layer (ONL), photoreceptor inner segments (ILS), photoreceptor outer segments (OLS), and retinal pigment epithelium (RPE).

2.3 Deep Learning: Artificial Neural Networks

Artificial neural networks (ANNs) represent a computational paradigm inspired by the intricate neural connections within the human brain [23]. Given the recent advancements in computer technology and the expansion of databases, these networks have gained immense popularity in the field of artificial intelligence, particularly in the context of deep learning, due to their remarkable ability to learn patterns, make predictions, and other complex tasks, such as automatic speech recognition, visual object identification, and object detection [24].

Deep learning, a subset of machine learning, revolves around employing deep architectures, characterized by multiple layers of processing. Learning can be supervised or unsupervised.

Supervised learning involves the use of an external "supervisor", that provides the desired response (target) to the neural network (NN) for a given input. In this case, learning occurs through the refinement of a model based on a set of examples rather than relying on rigid rule sets. The initial model undergoes successive updates as it learns from

evaluated instances, gradually achieving optimal performance in addressing the specific problem.

On the other hand, unsupervised training does not involve a target output. Hence, the network itself must be able to extract relevant features from the inputs and categorize them without the need for explicitly provided target labels or output information [25].

Another important machine learning concept is transfer learning where a pre-trained model is adapted or tuned for a related yet distinct task. The goal is to use the model's existing knowledge in an alternate domain to enhance its learning and generalization capabilities. This can be achieved by taking the pre-trained model and retraining it with a new dataset that is suitable for solving the new problem.

As mentioned, Neural Networks originated from the attempts to artificially model neural processes. From a computational perspective, it can be said that a neuron processes one or, often, multiple inputs to generate an output. The neuron is considered a fundamental unit for information processing. The artificial neuron is a logical-mathematical structure that seeks to simulate the form, behavior, and functions of a biological neuron, representing the elementary component of a neural network. When combined, these units form structures called layers which in turn construct the neural network.

A biological neuron receives an impulse via its dendrites and transmits a signal along its axon. Each neuron's axon possesses branches that interconnect with the dendrites of other neurons through synapses.

Roughly speaking, according to the artificial neuron model [26], as shown in Figure 2.3, the dendrites are the inputs (x_i), and their connections to the artificial cell body are established through communication channels associated with specific weights (w_i - simulating synaptic strengths), which are multiplied by the inputs. The stimuli received by the inputs are multiplied by each corresponding synaptic weight and then processed by the sum function, through a weighted sum, generating a certain level of activity. If this level of activity exceeds a certain threshold, the processing unit will produce a specific response as an output (y_i), simulating the axon. The firing threshold of the biological neuron is replaced by the activation function. Mathematically, this model can be represented as

$$y(x) = h(w_0 + \sum_{i=1}^{n} w_i x_i) \ , \tag{2.1}$$

where y_i is the output, x_i the inputs from the other neurons, w_i is the weight of the connection (input) x_i, h is the activation function, and w_0 is the bias.

Besides the network inputs or outputs from other neurons, each neuron is stimulated by a constant polarization called bias, w_0 or b.

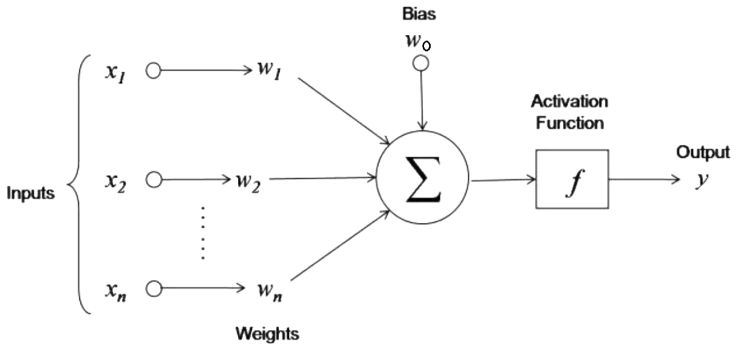


Figure 2.3: Artificial neuron model where inputs are acted upon by weights and summed to bias and lastly passes through an activation function to produce the final output [27].

During the learning process of the network, the training data is then used to adjust the learnable parameters, the weights w and bias w_0, so that the resulting function presents the best performance for the task at hand.

The figure above is an example of a Perceptron or Single Layer Perceptron (SLP) [28]. A SLP consists of only the input layer, the net sum and the activation function, and the output layer. This makes it suitable for solving linearly separable problems but limits its ability to handle more complex patterns that require hierarchical representations, which are typically addressed by multi-layer perceptrons (MLPs) [29].

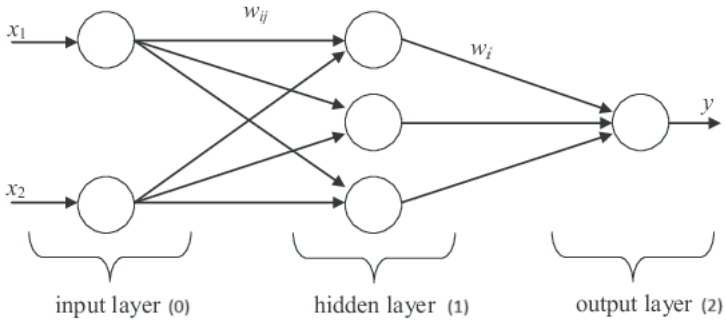


Figure 2.4: Diagram of a multi-layer perception [30].

In an MLP (Figure 2.4), there are hidden layers between the input and output layers. With the concept of the artificial neuron model in mind, the output $y(x)$, but also the set of transformed inputs $\phi_1(x), ..., \phi_m(x)$ that form a hidden layer can be defined as

$$\phi_i(x) = h_1(w_{i0}^{(1)} + \sum_{j=1}^{l} w_{ij}^{(1)} x_j) \text{ and } y(x) = h_2(w_0^{(2)} + \sum_{i=1}^{m} w_i^{(2)} \phi_i(x)) \ , \tag{2.2}$$

where the superscripts represent the layer number, $w_{i0}^{(1)}$ represents the bias term for the i-th neuron in the hidden layer (layer 1), $w_{ij}^{(1)}$ represents the weight of the connection between the j-th input neuron and the i-th neuron in the hidden layer, $w_0^{(2)}$ represents the bias term for the output neuron, in layer 2, and $w_i^{(2)}$ represents the weight of the connection between the i-th neuron in the hidden layer and the output neuron. The number of neurons in the hidden layer is m and the number of input features in the input layer is l.

If the number of hidden layers is 1 we have a shallow NN, whereas if we have more than 1 hidden layer we have a deep NN. It is a feedforward network meaning the current layer is fed only by the previous layer. The number of neurons on the output layer depends on the number of intended outputs for each specific problem. For a classification task, the number of output neurons is the number of classes.

Network Training

To optimize the output over time, the model has to be trained to modify the network's learnable parameters, weights and biases. This training phase can start after the model's architecture has been defined and the initial set of weights and biases has been selected. The network architecture defines, among other parameters, the type of training the network will undergo, enabling it to solve the intended problem.

The training of a neural network is usually performed using the backpropagation algorithm, which consists of a forward pass and a backward pass. In the forward pass, the output values are calculated and compared with the expected output, and the error is calculated. In the backward pass, this error is used to adjust the network's parameters with the aim of minimizing its value. This adjustment process continues by repeating

these steps until the error is minimized or at a reasonable value for the user, at which point the network is deemed to have acquired knowledge and is considered trained [31].

Starting from the basic objective of a neural network, which is to learn a non-linear function that relates N input-output vector pairs $(\boldsymbol{x}_k, \boldsymbol{y}_k)$, such that the following equation holds true

$$\boldsymbol{y}_k = h(\boldsymbol{x}_k), k = 1, 2, ..., n \ , \tag{2.3}$$

where k corresponds to a specific data sample and n is the total number of data samples. However, regardless of the application, most of the time the NN performs an approximation:

$$\tilde{\boldsymbol{y}}_k = h(\boldsymbol{x}_k) \approx y_k \tag{2.4}$$

The smaller the error resulting from the approximation in Equation (2.4), the better the network will map the problem. Consequently, the network's objective is to minimize the error for all input-output pairs, through a potential error function, the cost or loss function (L) that will optimize the approximation.

Let the vector $\mathbf{w}$ denote the set of the weights of the network. The loss function will be the expected value of the error ε_k between the real output $\boldsymbol{y}$ and the network's output $\tilde{\boldsymbol{y}}$ for each pair $(\boldsymbol{x}_k, \boldsymbol{y}_k)$:

$$L(\mathbf{w}) = E\left(\varepsilon_k\right) = E\left\{\|\tilde{\boldsymbol{y}} - \boldsymbol{y}\|^2\right\} \ , \tag{2.5}$$

where ε_k is given as the squared Euclidean distance between outputs. However, different problems and situations may require different error calculations or loss functions.

We can determine the optimal weights that minimize the loss function. One way of proceeding with this minimization is considering the gradient descent (GD) optimizer [32]. From an initial guess in the weight space, we take a small step in the direction of maximum decrease; i.e. from step τ to $\tau + 1$, likewise

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla L\left(\mathbf{w}^{(\tau)}\right) \tag{2.6}$$

$$\Delta \mathbf{w} = -\eta \frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} \tag{2.7}$$

where $\eta > 0$ is the learning rate and $\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}}$ is the gradient of the loss with respect to the weights. The GD method is useful with a sequential method, where we update $\mathbf{w}$ for each training event or iteration.

The learning rate is a hyperparameter that determines the step size at which a model's parameters are updated during training. A small learning rate results in slow convergence because the model takes small steps and may require a large number of iterations to reach the optimal or near-optimal solution. However, it is less likely to overshoot the minimum of the loss function. A large learning rate can lead to faster convergence, but it's also more likely to overshoot the optimal solution and may fail to converge or even diverge. This can lead to unstable training [33].

To compute all the derivatives required for the gradient descent minimization, to solve Equation (2.8), we use an algorithm called Error Backpropagation or "backprop" [31, 34], where the derivatives of the errors are propagated backward through the network. As seen, according to Figure 2.4, in a forward pass, the output can be written $y(\boldsymbol{x}) = h(u(\boldsymbol{x}))$ where

$$u(\boldsymbol{x}) = \sum_{i=0}^{m} \mathbf{w}_{1i}^{(2)} \phi_i(\boldsymbol{x}) \text{ and } \phi_i(\boldsymbol{x}) = h(\sum_{j=0}^{l} \mathbf{w}_{ij}^{(1)} \boldsymbol{x}_j) \ , \tag{2.8}$$

where the superscripts represent the layer number, $\mathbf{w}_{1i}^{(2)}$ represents the weight of the connection between the i-th neuron in the hidden layer and the output neuron, $\mathbf{w}_{ij}^{(1)}$ represents the weight of the connection between the j-th input neuron and the i-th neuron in the hidden layer. The number of neurons in the hidden layer is m and the number of input features in the input layer is l. Also $\phi_0 = x_0 = 1$. The sums are written over the nodes in the preceding layers starting from 0 to include the offsets (biases) which allow the network to learn them.

To evaluate the derivative of the loss function L with respect to the weight $\mathbf{w}_{1i}^{(2)}$, the chain rule can be applied

$$\frac{\partial L}{\partial \mathbf{w}_{1\mathrm{i}}^{(2)}} = \frac{\partial L}{\partial u(\boldsymbol{x})} \cdot \frac{\partial u(\boldsymbol{x})}{\partial \phi_i(\boldsymbol{x})} \cdot \frac{\partial \phi_i(\boldsymbol{x})}{\partial \mathbf{w}_{1\mathrm{i}}^{(2)}} \ , \tag{2.9}$$

where the first term is the derivative of the loss with respect to the network output $u(x)$ and depends on the choice of the loss function, the second term is the derivative of $u(x)$ with respect to the hidden neuron activation $\phi_i(x)$, and the third term is the derivative of $\phi_i(x)$ with respect to the weight.

Depending on the choice of activation and loss functions, these derivatives will be computed, and the result will be used in the weight update step of the GD optimization algorithm (Equation (2.8)).

Equations (2.8) and (2.9) consider weights associated with the connections between the second layer and the first layer, $\mathbf{w}_{1i}^{(2)}$. In practice, the backpropagation algorithm starts from the end of the network and recursively applies this same chain rule and thought process along the network backward, to determine all the gradients and update the weights until the input layer is reached. Furthermore, although this algorithm has been exemplified on an MLP with one hidden layer for simplicity, it works the same way for larger networks with L hidden layers, from layer *L+1* to L, and so on.

Loss Functions

Loss functions, also known as cost functions or objective functions, play a pivotal role in training and optimizing machine learning models. They evaluate the model's predictions by quantifying the disparity between the predicted values and the actual target values. By tuning the model's parameters, the objective is to minimize this function, thus guiding the learning process toward improving the model's performance.

In classification tasks, a loss function can be written as a sum, over all classes, of an error defined for each class i separately:

$$L(\mathbf{w}) = \sum_{i=1}^{m} L_i(\mathbf{w}) \ , \tag{2.10}$$

where $\mathbf{w}$ is a vector of weights.

For taking average loss instead of the sum, Equation (2.6) can be divided by the number of samples in the training dataset [34].

Commonly used loss functions for classification problems include:

- **Categorical Cross-Entropy Loss (CE)**:

  The Categorical Cross-Entropy (CE) loss or Softmax loss measures the difference between predicted class probabilities and the true class labels for each data point as follows:

$$CE\ Loss = -\sum_{j=1}^{n}\sum_{i=1}^{m} y_{ji} \cdot \log\left(\tilde{y}_{ji}\right) \tag{2.11}$$

  where y represents the true class labels in one-hot encoded form, $\tilde{y}$ represents the predicted class probabilities for all classes, *log* denotes the natural logarithm, m is the number of classes, and n the number of training samples.

  The categorical cross-entropy loss encourages the model to assign high probabilities to the true class labels while minimizing probabilities for incorrect classes. It's suitable for scenarios where each input data point belongs to exactly one class out of multiple classes [35, 36].

  The categorical cross-entropy loss is widely used in NNs in conjunction with the softmax activation function in the output layer of neural networks (NNs) for multi-class classification tasks [37].

- **Binary Cross-Entropy Loss (BCE)**:

  Applied to binary classification problems, this approach involves input data being categorized into one of two predefined classes by the model, being a special case of CE loss for a number of classes equal to 2. It measures the difference between predicted class probabilities and the true binary labels for each data point. Mathematically [36]:

$$BCE\ Loss = -\sum_{j=1}^{n} \left(y_j \cdot \log\left(\tilde{y}_j\right) + \left(1 - y_j\right) \cdot \log\left(1 - \tilde{y}_j\right)\right. \tag{2.12}$$

where y represents the true class labels, $\tilde{y}$ represents the predicted class probabilities for both classes, *log* denotes the natural logarithm, and n is the number of training samples.

The choice of a loss function depends on the nature of the task and the characteristics of the data. Picking an appropriate loss function is crucial for guiding the learning and optimization process and achieving the desired model performance since the backpropagation will be computed on this function and the resulting gradients are used to perform the parameters update.

Activation Functions

Activation or transfer functions introduce non-linearity to the model's transformations and determine whether a neuron should be activated or not based on the weighted sum of its inputs. Without activation functions, the network would be limited to linear operations, severely restricting its capacity to learn complex patterns or parameters within data. Given this, they enable neural networks to generalize across diverse datasets and model a wide range of functions, making them capable of approximating both linear and nonlinear relationships between inputs and outputs [38, 39, 40].

These functions are usually monotonic and differentiable and restrict the amplitude of the neuron's output signal. There are several types of activation functions, with some of the most frequently employed being:

- **Sigmoid or Logistic**:

  The sigmoid function is a smooth and continuous function that maps any real input value to a value between 0 and 1. With an input of x, the formula for the sigmoid function is [39]:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.13}$$

  Its output can be interpreted as a probability of belonging to a certain class based on some input features, so this function is especially used for models where the

intended output is a probability. It has been widely used for binary classification tasks, where the expected prediction is a probability for a binary output (0 or 1).

However, this type of function tends to saturate at 0 or 1 as the input becomes very large or very small. This results in gradients that are extremely small, a phenomenon known as vanishing gradients, which can interfere with the learning process during backpropagation. This can slow down the training process and make it challenging for neural networks to learn effectively, particularly in deep networks with numerous layers.

- **Hyperbolic Tangent (tanh)**:

  The hyperbolic tangent function is also a continuous function that maps any real input value to a value between -1 and 1 [39], according to the following formula, for an input of x:

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{2}{1 + e^{-2x}} - 1 \tag{2.14}$$

  Similarly to the sigmoid function, the tanh function is also used in binary classification tasks, with the advantage of being zero-centered, meaning that the average output is close to zero, making it well-suited for optimization algorithms that rely on symmetric gradients. This might help with the convergence of the neural network during training.

  Additionally, it can model stronger non-linearities, since this function allows for negative correlation in comparison to the sigmoid, which only gives varying degrees of positive correlation. Nevertheless, the tanh function also suffers from the problem of vanishing gradients for large input values.

- **Rectified Linear Unit (ReLU)**:

  The Rectified Linear Unit function maps any negative input to zero and leaves any positive or zero inputs unchanged, having a range of $[0, +\infty[$ [39]. The mathematical expression representing the ReLU function, for an input x, is as follows:

$$f(x) = max(0, x) \tag{2.15}$$

Due to its mathematical straightforwardness, it becomes more computationally efficient within deep learning architectures, since its simpler operations improve the convergence, resulting in faster learning. This function has gained widespread popularity in recent times, specifically in hidden layers of large-scale neural networks [41].

Another benefit of the ReLU function is that it is less susceptible to the vanishing gradient issue when dealing with positive input values. However, it can suffer from the problem of dead neurons, where negative inputs that output zero fail to contribute to the learning process. This occurs as the resulting graph does not map these negative values appropriately. Consequently, this phenomenon can lead to a reduction in model performance, given that its ability to fit or train the data will be decreased. This issue can be fixed using a proper learning rate.

- **Leaky ReLu**:

  The Leaky ReLU function [42] is a variant of the ReLU function that addresses the problem of the dying ReLU. It introduces a small negative slope for negative input values (leak), instead of setting them immediately to zero, increasing its range to $]-\infty, +\infty[$. The formula for the Leaky ReLU function, for an input of x, is:

$$f(x) = \begin{cases} x & x > 0 \\ ax & otherwise \end{cases} \tag{2.16}$$

  where a is a small positive constant, usually set to 0.01 when it's not randomized or 0.2.

- **Softmax**:

  The Softmax function is commonly used as the final activation function in the output layer of NNs for multi-class classification tasks, where the goal is to predict the

probability or likelihood of each class given some input features [43]. This function maps a vector of real scores (logits) to a probability distribution across the different classes, like a vector of probabilities that add up to one. It can be represented mathematically as follows, for a vector of logits z:

$$f(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{m} e^{z_j}} \tag{2.17}$$

where z_i is the i-th element of the input logit vector z for all classes, m is the number of classes, and $f(z_i)$ is the output probability of belonging to class i. It can also be applied for a logit matrix.

The softmax function magnifies the differences between logits, making the class with the highest score stand out more distinctly. In situations like training a neural network for image classification, the softmax function enables the network to make confident predictions by selecting the class with the highest probability.

Optimizers

As mentioned previously, optimizers are algorithms used during the training phase to update the model's learnable parameters, by finding the optimal set of parameters that minimize the loss function. There are several optimization algorithms available in deep learning, each with its own advantages and disadvantages:

- **Stochastic Gradient Descent (SGD):**

  The stochastic gradient descent (SGD) [44] is a widely used optimization algorithm that, similarly to the GD, updates the model parameters based on the gradient of the loss function with respect to the parameters (Equation (2.6)).

  It is also an iterative method. However, unlike gradient descent, it estimates the error regarding a single random example of the training set. The weights are updated after each iteration, meaning that the loss function is tested on a training sample and the model is updated.

In the regular GD method, the weights are updated based on a gradient computed over the entire training set, which slows down the process. It also requires a large amount of memory to store this temporary data, making it a resource-hungry process.

The SGD addresses both memory constraints and the high time cost of running the backpropagation algorithm over the entire training set, but it still doesn't suffice, since the error estimate with respect to only one training example at a time won't provide reliable approximations. The frequent updates result in convergence to the minimum in less time, but it comes at the cost of increased variance that can make the model overshoot the required position.

Addressing this issue, the mini-batch gradient descent [45] uses a fixed size subset (mini-batch) of the training data to estimate the gradient and compute the error. Usually, the term SGD is also used for this last batch-based algorithm. Formally, both algorithms can be written as:

$$w^{(\tau+1)} = w^{(\tau)} - \eta \nabla L_j \left(w^{(\tau)} \right), \tag{2.18}$$

where τ is the iteration, and j = *1, 2,... , n* is the data point j (in the case of SGD) or the mini-batch j (in the case of mini-batch gradient descent).

Therefore, this last optimizer will update the weights every iteration, but with respect to a batch of samples instead of only one data point. The batch size is a hyperparameter of the NN. The mini-batch GD benefits from the stochasticity of the SGD while being less noisy due to the averaging of gradients over the batch, leading to a more stable and efficient training.

- **Adaptive Gradient Algorithm (Adagrad):**

  Based on the previous method, the Adagrad is an adaptive learning rate optimization algorithm that adapts the learning rate to the individual parameters of the model based on the historical information of gradients [46].

It accumulates the squared gradients for each parameter and divides the learning rate by the square root of the sum of the gradients, decaying the learning rate in proportion to the updated history of the gradients, as such:

$$w_{t+1,i} = w_{t,i} - \frac{\eta}{\sqrt{G_{t,i} + \epsilon}} \nabla_w L\left(w_i\right) \tag{2.19}$$

where i is the index for the elements in w, $G_{t,i}$ is a diagonal matrix with the non-zero elements equal to the sum of squares of the previous gradients of w up to the time point t, and ϵ is the smoothing term, usually between 1×10^{-4} and 1×10^{-8}, that prevents zero division.

Thus, parameters associated with larger accumulated gradients have smaller effective learning rates, while those with smaller accumulated gradients have larger effective learning rates. If the parameters change frequently (higher variance), the learning rate, too, changes frequently.

This optimizer is particularly effective for handling sparse data, where some features may have much larger gradients than others, and dealing with features with different scales, since it alters the learning rates according to the input provided. Features with larger gradients will have smaller accumulated squared gradients over time and vice versa.

Even though the Adagrad can converge faster, a disadvantage of this approach is that the learning rate decays excessively and after some time it approaches zero. This may cause the model to stop learning, which is a bigger concern for deeper networks.

- **Adaptive Moment Estimation (Adam)**

  Adam [47] is another adaptive learning rate optimization algorithm based on the Adagrad and Root Mean Square Propagation (RMSProp) [48] optimizers.

  The introduction of this improvement aimed to overcome issues of the Adagrad like vanishing or exploding gradients, reducing the aggressiveness of the learning rate,

while keeping the adaptive learning rate feature intact.

Adam introduces a form of momentum-like updates into the optimization process, in which the gradient only directly affects the velocity. It adds a term that increases the velocity in directions with consistent gradients, which results in faster convergence. However, this method uses an exponentially decaying average of the first and second moments of the gradients, the mean, and uncentered variance, respectively, for each parameter, to adjust the learning rate. It stores these estimates for the moving average of the past gradients, m_t, and the past squared gradients, v_t. Parameters β_1 and β_2 control the decay rates of these moving averages, like so:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1)\,\nabla_w L\,(w) \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2)\,\nabla_w L\,(w)^2 \end{aligned} \tag{2.20}$$

where β_1 is the exponential decay rate for the first moment estimate and β_2 the exponential decay rate for the second moment estimate, with recommended values of $\beta_1 = 0.9$ and $\beta_2 = 0.99$. Using the physics momentum analogy, here $\beta_2 v_{t-1}$ is the momentum term, v_t is the updated velocity, and v_{t-1} is the prior velocity.

Besides this, the Adam optimizer also introduces bias correction during the initial iterations to correct any bias introduced by the estimates of the first and second moments, which may bias β_1 and β_2 towards zero:

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \end{aligned} \tag{2.21}$$

This correction is especially important when the optimization starts with small learning rates. Finally, these are then used to perform the weight updates as shown:

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t, \tag{2.22}$$

The Adam optimizer has consistently demonstrated exceptional performance when compared to other adaptive learning algorithms. As a result, it is frequently recommended as one of the default choices for optimizing machine learning models [49].

Overfitting

One of the most important obstacles to consider when training a network is overfitting [50]. It occurs when the model learns to perform exceptionally well on the data used to create it but fails to generalize effectively on unseen data, resulting in a low error rate (high accuracy) on the training data, but a high error rate on independent test data samples.

It can occur when a network becomes too complex (e.g. more nodes/layers) or is trained for too long (overtraining) and starts to memorize the training data instead of learning the underlying patterns. When building deeper models, as a network grows in layers, its decision boundary becomes increasingly more flexible, but the classifier may conform too closely to the training points.

The overfitted model will "learn beyond the concept", capturing and learning from data noise (irregularities caused by sampling errors). Being overly adjusted to the training data makes it so the model cannot generalize or adapt well to variations or different data distributions.

This problem is trackable: if the fraction of misclassified events for test and training samples is monitored, by plotting the training and validation loss, as depicted in Figure 2.5, it will usually decrease for both as the boundary is made more flexible. When the error rate for the test sample starts to increase, with respect to the one for the training sample, we will observe overtraining. A gap between the training and validation will start to appear and it reflects the amount of overfitting. The flexibility of the boundary is optimum at the minimum of the error rate for the test sample, just before its loss starts to increase. The training can be stopped at this point (early stopping).

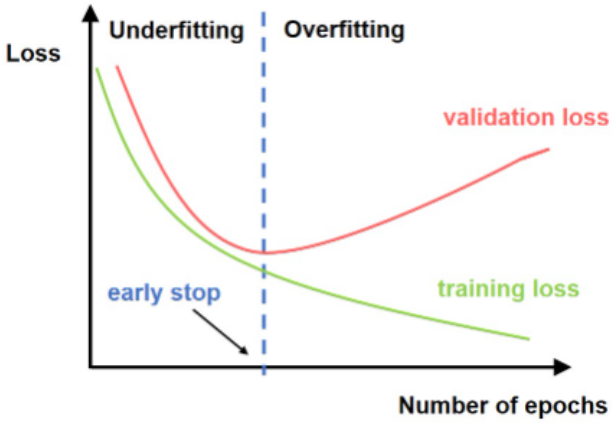


Figure 2.5: Example plot of training and validation loss monitoring per epoch. The green curve represents the training loss and the red curve represents the validation loss. After several epochs, the validation loss starts to increase while the training loss is still decreasing. The blue line represents the optimal loss minimum for the validation sample where early stopping should be performed [51].

Therefore, addressing overfitting is crucial for building robust models and preventing this issue. A way of doing this is by applying regularization techniques such as dropout.

Dropout

Regularization methods consist on adding a penalty or constraint in the model's training process. These constraints discourage the model from learning complex features that are only relevant towards the training data.

Dropout [52] is a regularization technique commonly used in deep neural networks to prevent overfitting and improve generalization, by reducing the reliance on specific neurons or features.

Through this strategy, random nodes inside the model's layers are temporarily removed during training or "dropped out", meaning the output of those particular nodes is set to zero for that specific training iteration. Therefore, it results in the training of a distinct sub-network using the standard backpropagation algorithm, during each training step. Each subnetwork corresponds to a different combination of active and deactivated neurons and by averaging the predictions of these subnetworks during inference, dropout approximates the ensemble effect, which can lead to improved generalization.

The fraction of deactivated nodes in each layer is the dropout rate hyperparameter set usually between 0.2 and 0.5.

This technique helps prevent overfitting, by forcing the model towards learning more robust features that don't depend on the specific nodes that are active during training, which may belong to noisy or small datasets. Besides this, dropout can make the model less sensitive to hyperparameter choices like the learning rate. Below is a scheme for the dropout method (Figure 2.6).

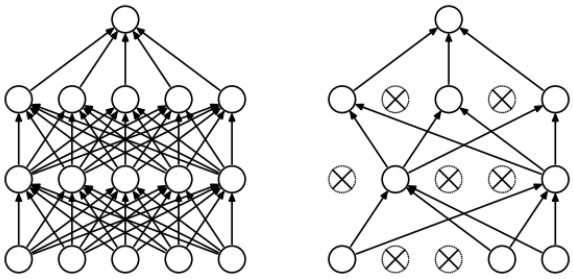


Figure 2.6: An example of the application of dropout [52]. The image to the left represents a standard neural network and the image to the right represents the same neural network with dropout.

2.3.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) can be formally defined as a class of neural network architectures that employs convolutional operations in at least one of their layers, contrasting with the conventional matrix multiplication approach [24].

These types of networks are specifically designed for tasks involving grid-like data, such as images and video. CNNs have been increasingly used for pattern recognition, image classification, object detection, image segmentation, semantic segmentation, and other analogous tasks driven by their promising results and the potential to improve the accuracy and efficiency of algorithms developed for those purposes [53, 54].

The overall behavior of a Convolutional Neural Network (CNN) closely resembles that of a standard NN. Therefore, many of the components and working principles discussed throughout this chapter regarding Neural Networks remain applicable to a CNN.

Nevertheless, there are notable distinctions, mainly in its structure, worth emphasizing in this subsection.

Figure 2.7 depicts a schematic of a typical CNN.

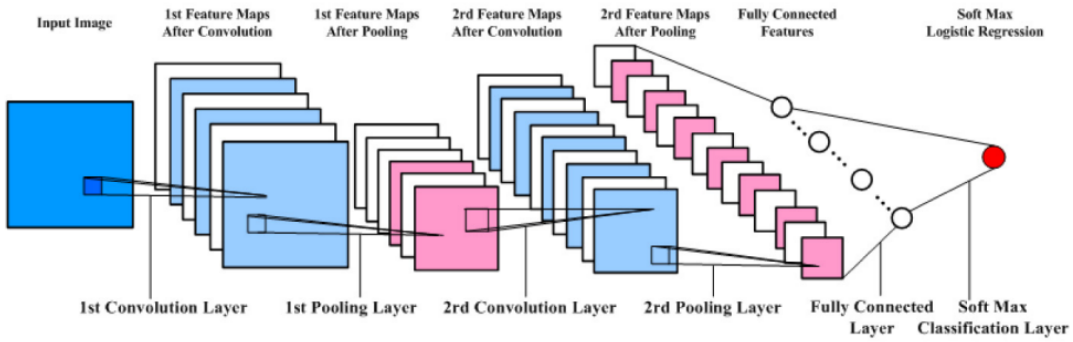


Figure 2.7: Scheme for a typical convolutional neural network architecture [55].

The input to a CNN is typically a 2D image, translated to an array of size *(width, height, number of color channels)*. The network is trained on a preferably extensive dataset of images for which the labels, the ground truth, from various images are known. Ground truth is information about the images that is known to be real or true. During the training process, the algorithm learns to extract increasingly complex features (parameters) from the images, in the convolution layers, and ultimately predicts class scores or probabilities using those learned features in its fully connected layers. CNNs have a hierarchical structure for this feature extraction to learn the spatial relationships and patterns (edges, textures, shapes, ...) of the images.

Essentially, the basic pipeline for a CNN involves performing convolution on the input image, accompanied by an activation layer to allow non-linearity, to get an activation map. An activation map refers to the result of applying an activation function to a feature map, which refers to the output of a specific layer in a CNN, commonly a convolutional operation. They are enhanced versions of feature maps, as they emphasize which parts of the feature map are activated (have non-zero values) and which are not. Then, a pooling layer is applied to make the model more robust. Finally, the outputs from this last layer are passed to a fully connected layer, that will provide a probability map for each intended class. This process is repeated until the trained weights are well-defined and all features detected [24, 56].

Convolution Layer

In convolution layers, a set of learnable filters (kernels), which are smaller-sized weight matrices, is applied to the input data. In the forward pass of a convolutional operation, each filter traverses across the input volume, computing dot products at each position. Specifically, the pixels within a local region (receptive field), around the position (x, y), are multiplied element-wise by the filter's weights. The resulting elements are summed up to compute the output value. This convolution process between the filter and the image pixel values is done for each spatial location across the input volume. Mathematically, this operation can be written as:

$$(I * K)(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} I(x - i, y - j).K(i, j) \tag{2.23}$$

where $(I * K)(x, y)$ is the output of the convolved image at position *(x,y)*, I is the input image, K is the convolutional kernel (filter), and i and j are the indices used for the summation, which typically range over the filter's dimensions.

After each filter is applied, the result of the 3D input is a 2D feature map, that combined with the other filters' outputs form the 3D final output to the convolution layer. Hence, a feature map represents the response of one particular filter to the input data, capturing the specific patterns of features [24, 56]. Figure 2.8 represents an example of the convolution operation.

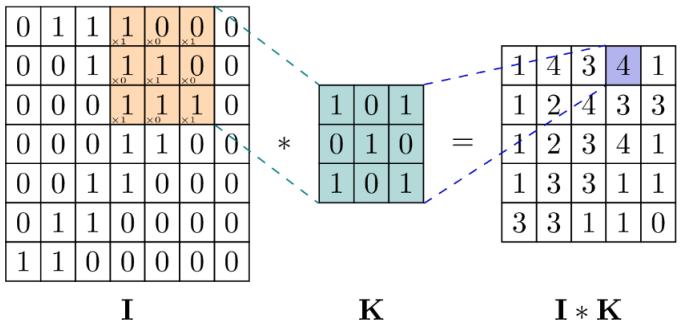


Figure 2.8: Pictorial example of the convolution operation with a kernel size of 3×3. The kernel and an equal size image sub-matrix is extracted. The dot product of the matrices is saved in the output matrix. The filter then moves by the number of pixels according to a defined stride and the dot product calculation is repeated [57].

The dimensions of the filter matrix represent a hyperparameter of the CNN, determining how many pixels or elements the filter covers in each dimension of the input. The receptive field's size will evidently be the same as the kernel's defined size. Common filter sizes are 3×3, 5×5, and 7×7. Besides this, another hyperparameter is the number of filters used. In the initial convolutional layer(s), it often starts relatively low (e.g. 32, 64) and gradually increases in deeper layers (e.g. 128, 256). The number of filters can be increased depending on the complexity of the data or task if it involves capturing more complex features.

The weights in the filter matrix are updated every time backpropagation is performed during the learning process, in each learning iteration over the training set.

Finally, an activation function is applied element-wise to introduce non-linearity into the model. This enables the network to learn complex, non-linear relationships within the data. Effectively, the convolution layer will capture the local patterns and features in the image data, and trying to learn from them (feature extraction).

It's important to highlight an optional hyperparameter employed in CNNs: the stride. The stride denotes the pixel-wise step size at which the convolutional filter (kernel) slides across the input data during the convolution operation. For example, a stride equal to 2 halves the dimensions (downsampling). Larger strides have the effect of diminishing the spatial dimensions of the output feature map as the filter covers a smaller area with each step [24, 56].

Padding

Padding is an important operation that affects the spatial dimensions of the output feature maps that can also be used in convolutional layers.

Padding refers to the process of adding extra values, usually zero (Zero-Padding), around the border of the input matrix before applying a convolution operation. It can be employed to control the size of the output feature maps produced by convolutional layers. Zero-Padding or same padding keeps the size constant, so the input and output have the same dimensions after the convolution, to preserve the input volume's size.

This operation impacts the output feature map's size in the following manner:

$$output\ size = \frac{N + 2P - F}{S} + 1 \tag{2.24}$$

where N is the input size, S is the stride, and F is the filter size and P the amount of padding (in pixels) [24, 56].

Pooling Layer

Pooling layers reduce the spatial dimensions (width and height) of the feature maps, which helps reduce the computational load and control overfitting by focusing and retaining important features. It is applied to each feature map independently, meaning that the depth dimension remains unchanged. A consistent depth simplifies the network's architecture and makes it easier to transition from one type of layer to another. There are various pooling methods, such as max pooling, average pooling, and global pooling.

The most widely used pooling technique is max pooling. When a max pooling filter is applied, a small square or rectangular window slides over the input feature map. At each position of the window, the maximum value (strongest activation) within it is selected and retained in the output feature map.

This layer is usually applied in between convolutional layers and takes two hyperparameters: size, which determines the window area of every pool (often 2×2, 3×3), and stride. The output size is given by:

$$output\ size = \frac{N - F}{S} + 1 \tag{2.25}$$

where N is the input size, S is the stride, and F is the filter size [24, 56]. The image below provides a visual representation of the max pooling process for a better understanding of the concept (Figure 2.9).

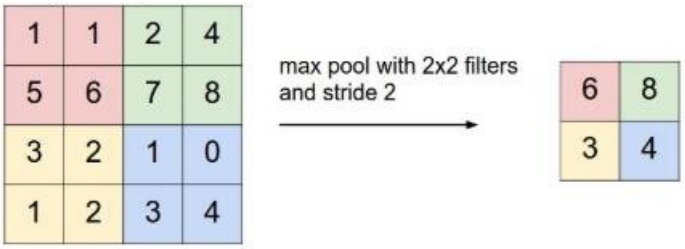


Figure 2.9: Pictorial example of the max-pooling operation with a kernel size of 2×2 and stride of 2 [58].

Fully-connected Layer

Fully connected (FC) layers (Figure 2.10) are traditional neural network layers where each neuron is connected to every neuron in the previous layers (dense connectivity) and subsequent layers, if any, as they are frequently used as the final layers of CNNs. They aggregate and interpret the features extracted by earlier layers by making predictions based on these high-level features.

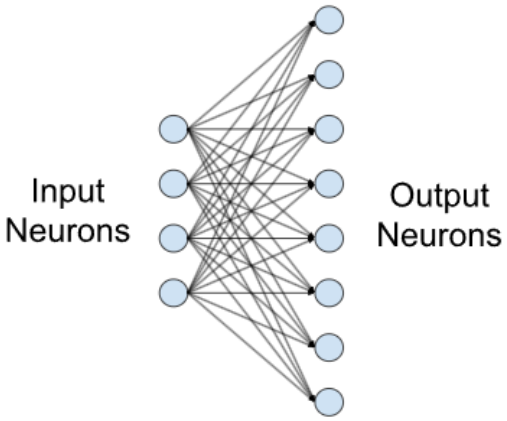


Figure 2.10: Diagram illustrating a fully-connected layer, where each output neuron is "fully connected" to all the input neurons [59].

An FC layer consists of a collection of neurons or units that have associated weights for each of its connections to the previous activation units. Each neuron applies an activation function to the weighted sum of its inputs, according to Equation (2.1). When using this type of layer as the final network layer, the associated activation function may differ from the others throughout the network. As explained in the last subsection, selecting an activation function depends on the task in question. Usually, in FC layers, the final activation function for multi-class classification is the Softmax function, while Sigmoid

activation is used for binary classification.

Thus, the output of a fully connected layer is a set of values (activations), one for each output neuron in the layer, and can be thought of as the learned features or representations of the input data. The number of neurons in the final FC layer typically corresponds to the number of classes in a classification task [24, 56].

A few key characteristics of Convolutional Neural Networks were demonstrated throughout this subsection. Firstly, even though convolution limits the connectivity between neurons in space through the use of kernels, this constraint is held over the entire depth of the input volume (depth-wise connectivity), allowing it to cover all parts of the input considering information and capturing features across all channels or sections in the input volume.

Furthermore, in CNNs, each neuron in a convolution layer is connected only to a small local region of the previous layer (sparse connectivity), defined by the size of the receptive field of the kernel. Therefore, a local input has an impact only on that local's output. This attribute of the CNNs allows for the hierarchical feature learning from input data.

In parallel, the sparse connections allow the network to focus on local patterns and features, making it robust to changes in object position within the image. This consistency is achieved by sharing weights (parameters) across the local connections. As a result, they are able to achieve translation invariance with respect to the classification output, meaning that regardless of the exact position of an object the classification should be the same.

In traditional neural network layers, all inputs have an impact on all outputs, since dense connectivity is dominant across all layers. This dense connectivity results in a large number of parameters of the network and an excessive computational burden, especially as the network grows in size. On the contrary, the sparsity in CNNs reduces the number of weights (weight sharing) and computations required in the network, making them computationally efficient and the most appropriate tool for handling image datasets [24, 56].

Chapter 3

Methods for Retinal Layer Segmentation

Retinal layer segmentation identifies and separates the different retinal layers in medical images and is used in OCT analysis tasks. It is a necessary process as it allows the detection and monitoring of various conditions [9].

Consequently, various systems have been developed over the years to find increasingly better solutions to automate the segmentation of retinal layers. In this respect, different attempts have resulted in a wide range of techniques, which will be analyzed below. The various methods developed to segment the retinal layers in medical images can be broadly divided into two categories:

(**A**) **Traditional Approaches**

These techniques are primarily based on traditional models of retinal boundary segmentation, taking into account the anatomical information and optical characteristics of each retinal layer. According to the principles on which they are based, this approach can be divided into the following sub-categories: pixel classification (*thresholding*) [60, 61], *active contours* [62, 63, 64, 65, 66], and 3D *graph search* [67, 68, 69, 70, 71, 72, 73, 74].

(B) Machine Learning Approaches

In addition to the above interpretations of the segmentation problem, artificial intelligence approaches (machine learning algorithms), such as pattern recognition techniques, can also be applied. Recently, the most widely used segmentation approaches are based on deep learning techniques such as convolutional neural networks.

Since this study is closely related to deep learning, as it proposes the development of a convolutional neural network for retinal layer segmentation, this chapter will focus on machine-learning approaches, particularly deep-learning approaches, to the retinal layer segmentation problem. Thus, traditional retinal segmentation will not be further addressed in this work.

3.1 Traditional Machine Learning Methods

Various machine learning methods based on artificial intelligence can be used to perform segmentation procedures on structures identified in medical images, by recognizing their patterns. These classification models can be supervised, where manually segmented training data is required and used as a reference to automate new cases, such as support vector machine (SVM) algorithms [75]. On the other hand, classification algorithms can also be unsupervised, where they don't need training data, but rather initial parameters to perform segmentation, like clustering techniques such as the fuzzy c-means (FCM) algorithm [76, 77].

Support vector machine algorithms aim to find the optimal hyperplane that maximizes the distance between data points in different classes. This hyperplane is chosen to minimize classification errors. Support vector machines (SVMs) use support vectors, which are the data points closest to the decision boundary, to determine the hyperplane that best separates classes [78].

Fuller et al. [75] used a multi-resolution hierarchical support vector machine method to build a semi-automatic segmentation system that considered scalar intensity, gradient, spatial location, neighborhood mean, and variance over multiple resolutions, allowing it to deal with noise. It also considers a voxel's mean value. The proposed SVM classifier was able to segment healthy and diseased retinas. However, the method requires the clinician or user to color the areas of interest in the volume, which are used as the training data for the method. As more regions are painted by the user, the dataset grows and the SVM also becomes more complex, resulting in an increasingly longer runtime that precludes its application. It was reported that 68% of the thickness differences between the SVM segmentation and the ground truth was under six voxel units.

The FCM algorithm is a powerful clustering technique. FCM assigns data points to clusters with degrees of membership, allowing for more flexible and nuanced clustering solutions. It calculates membership values for each data point, quantifying the degree to which it belongs to each cluster. By iteratively refining cluster centroids based on these fuzzy memberships, FCM efficiently handles data with overlapping or uncertain cluster assignments [79].

In 2008, Mayer et al. [76] proposed an automatic clustering technique using the FCM algorithm to calculate the thickness of the RNFL layer in B-scans of healthy retinas and retinas of eyes diagnosed with glaucoma. The proposed FCM algorithm clusters the data into multiple categories and identifies and separates the RNFL from the other layers according to pixel intensity values. Geometry corrections were also considered to deal with any geometric distortions in OCT images to avoid errors when calculating thickness values. The authors reported that 97% of the upper and 74% of the lower RNFL layer boundaries were within two pixels of the ground truth on the test dataset.

More recently, traditional machine learning methods, such as SVM and FCM, allowed for a segmentation accuracy of up to 2 pixels [80].

3.2 Deep Learning Methods

Recently, advances in deep learning have given rise to increasingly efficient methods for tackling image segmentation. Mechanisms such as convolutional neural networks are robust tools that automatically learn complex representations from large amounts of data. They have been increasingly utilized for segmenting layers of the retina due to their promising results and potential to enhance the precision and effectiveness of algorithms [10].

CNNs are the most extensively used networks in image classification tasks, particularly for retinal layer segmentation challenges. They originated in 1980 with Kunihiko Fukushima's proposal of the *Neocognitron* [81] and have been continuously refined since. There are now a significant number of systems and effective learning methods available, enabling the problem at hand to be tackled from multiple angles.

One of the initial studies to present the possibilities of utilizing CNNs for solving intraretinal segmentation problems occurred in 2017, by Fang et al. [55]. The authors suggested a structure that merges a convolution neural network with the graph search method (CNN-GS) to automate the segmentation of nine retinal layer boundaries and thus eight retinal layers (RNFL, GCL + IPL, INL, OPL, ONL, inner segment ellipsoid (ISE), OS, and RPE and drusen complex (RPEDC)) in OCT images of patients diagnosed with age-related macular degeneration (AMD).

For this purpose, a modified CNN (Figure 3.1) for the Cifar database (Cifar-CNN) [82, 83] was used to extract features of each specific retinal boundary and train a classifier to generate class labels and probability maps, where a graph search method [69] was applied. In this last step, the final boundaries of the retinal layers in the OCT images were derived from the generated probability maps.

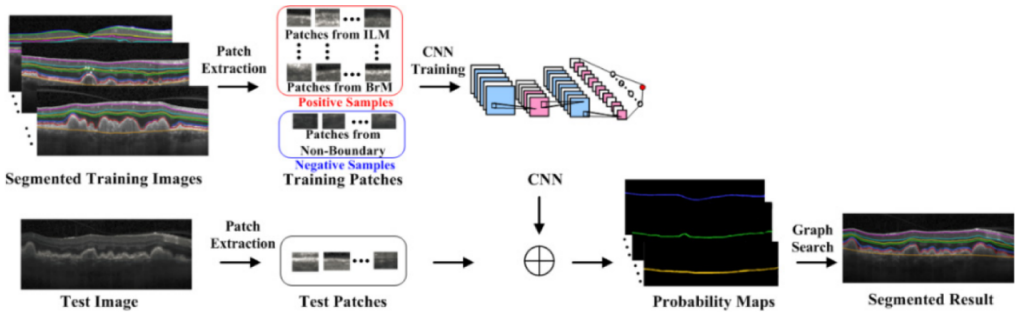


Figure 3.1: CNN architecture for the Cifar database [55].

Although this method manages to automate the segmentation of nine retinal layer boundaries to a certain extent, given that it obtains satisfactory results of a mean absolute error in segmentation between the manual and automated gradings of 1.26 pixels, it is sensitive to the position of the boundaries in the OCT images and the computational burden is quite high.

Among the many state-of-the-art CNNs that have been developed, **ResNet**, **DenseNet**, **U-Net**, and **RelayNet** are a few worth highlighting. Over the years, these networks have been applied in various ways to solve the problem of retinal layer segmentation. Throughout this subsection, some of the contributions and work developed based on these models will be explored, as well as the models' characteristics.

ResNet

A ResNet (Residual Network) [84] is a CNN that has residual blocks as a fundamental architectural component. In these blocks (Figure 3.2), the final output, *f(x)*, that is fed into the activation function will be the sum of the output from the convolutions and the input (identity function x). This introduces the residual learning concept where the layers inside the residual block will try to learn during the training a residual function $R(x) = f(x) - x$ instead of the output $f(x)$ directly, as regular CNNs do.

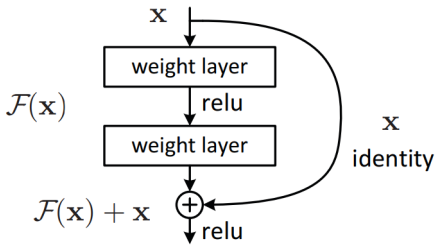


Figure 3.2: Residual block [84].

In 2019, Liu et al.[85] used a ResNet, as depicted in Figure 3.3, and an optimized structured random forest classifier [86, 87] to automate the segmentation of different retinal layers in OCT images.

Feature information was extracted by training the ResNet on the OCT images. These features were then used to optimize the structured random forest and accurately classify the different layers, predicting their boundaries by generating probability maps for each one.

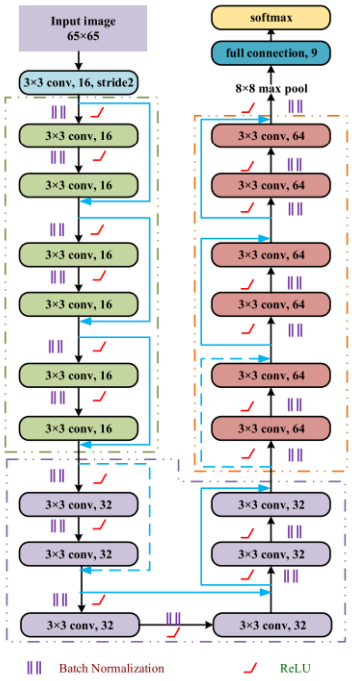


Figure 3.3: ResNet architecture used in Liu et al. [85].

The results obtained for this algorithm showed to be satisfactory, with a mean absolute error of 1.215 pixels and an F1-score of 0.885. The F1-Score ranges from 0 to 1, where a higher F1 score indicates better model performance (these metrics will be defined in

Chapter 4).

DenseNet

The DenseNet network [88] is similar to the ResNet architecture. However, its architecture comprises dense blocks, each containing several layers that will be connected to all the previous layers. Within a dense block, the output feature maps of each layer L are concatenated with the feature maps of all previous layers, according to:

$$f_L = H_L([f_0, f_1, ..., f_{L-1}]) \tag{3.1}$$

where f_L is the feature map of layer L, $[f_0, f_1, ..., f_{L-1}]$ is the concatenation of the feature maps generated in layers $0, ..., L-1$, and H_L is the activation function applied to layer L.

This implies that for L layers in a dense block, the output of each block will comprise L sets of feature maps concatenated together. Thus, for a typical network with L layers, there are L connections between the layers, while in a DenseNet with L layers, there will be about $L(L+1)/2$ direct connections. Figure 3.4 is a depiction of the DenseNet.

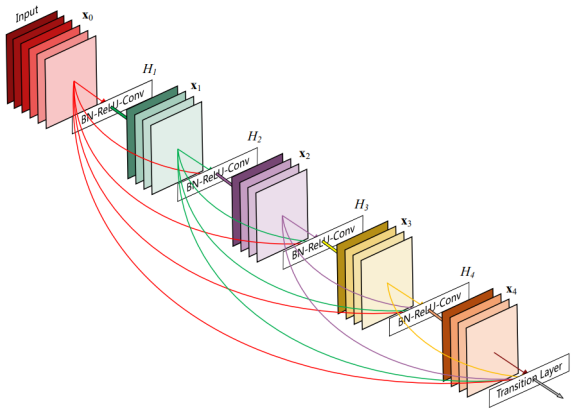


Figure 3.4: DenseNet architecture [88].

In Pekala al.[89], the authors present a deep learning approach to automatically segment retinal OCT images into the following retinal layers: pre-retinal space and NFL complex, NFL-GCL, IPL-INL, OPL-ONL, and Bruch's Membrane (BM) to Choriocapillaris complex. To accomplish this, they resort to a DenseNet (Figure 3.5) to segment pixels composing the layer.

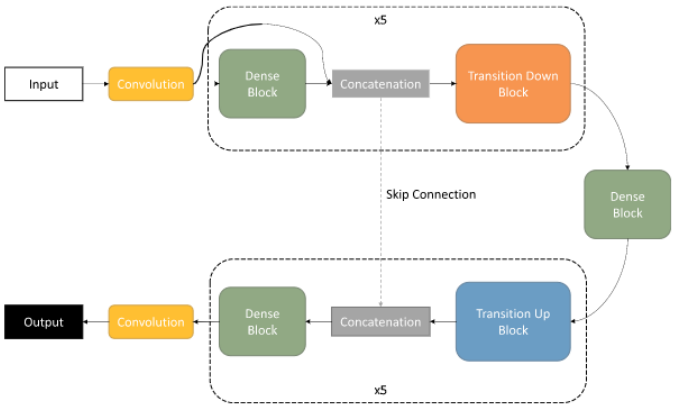


Figure 3.5: DenseNet architecture used in Pekala et al. [89].

The authors evaluate their method's performance by calculating the surface estimation accuracy, after which the DenseNet obtained a mean absolute error of 1.06 pixels.

A major problem in applying deep convolutional neural networks to medical image segmentation problems is the lack of labeled training data, which is a time-consuming and expensive process. In the approach taken by Sedai et al.[90], a semi-supervised uncertainty-guided student-teacher learning method is studied for the segmentation of eight retinal layers (RNFL, GCL+IPL, INL, OPL, ONL, IS, OS, RPE).

This model, U-SLS (Figure 3.6), has two segmentation networks of DenseNet architecture: the student network, responsible for learning an adequate representation of the data and the main segmentation task, and the teacher network, which controls the learning of the first network and models the unreliability of the segmentation predictions.

The training set for the segmentation network consists of a limited number of classified samples and a considerable number of unclassified images. First, the teacher model is trained with the labels using Bayesian deep learning [91]. The output is a segmentation map and an uncertainty map. The uncertainties, which correspond to the confidence values of the segmentation output of the teacher network, guide the selection of additional data to be annotated and incorporated into the training process of the student network. This iterative process continues until the model's performance reaches a satisfactory level.

The goal is to make the most effective use of the classified data to achieve accurate segmentation results.

According to the results, this U-SLS framework achieved, on average, a dice coefficient, which indicates the similarity between the obtained segmentation and the real segmentation, of 0.82 ± 0.07, being 1 the greatest similarity (metric defined in Chapter 4).

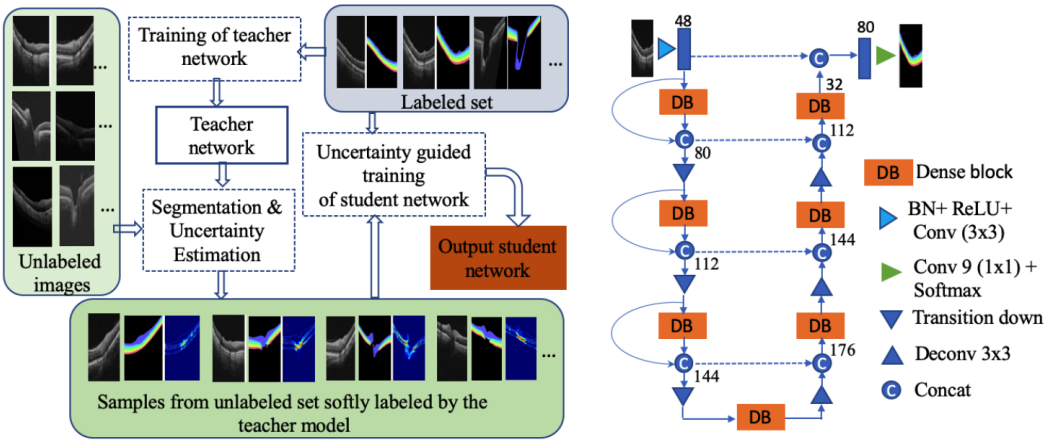


Figure 3.6: U-SLS architecture [90].

U-Net

The U-Net or U-Shape architecture, proposed by Ronneberger et al. [92], is a very popular type of CNN and the basis of many networks currently in use. It consists of two main parts: the encoder (contracting path), which consists of a series of convolution and max-pooling layers, reducing the spatial resolution of the input image and extracting the features, and a decoder (expanding path), which consists of transposed convolution layers that perform up-sampling on the previous representation to get the original resolution, mapping the extracted features in the output image. Thus, the encoders are used to learn a hierarchy of contextual features, and decoders are used to perform semantic segmentation, which is the process of assigning a label to each pixel in an image to indicate its class/category.

Connecting the encoder and decoder are skip connections that perform concatenation operations between the encoder's feature maps and the corresponding decoder layers, allowing the expanding path to access the representation of features at different resolutions/scales, thus preserving the fine details in the image.

The final layer of the U-Net is a 1×1 convolution layer used to adjust the format of the network's output, by matching the channels at the end of the decoder with the number of classes. In general, this last layer maps the final feature vector to the desired number of classes for classification tasks, transforming the learned features into the appropriate format for the final output segmentation map. For semantic segmentation, the output is a probability map of the same size as the original image, indicating the likelihood of the presence or absence of each object in the image, performing the final segmentation operation. Figure 3.7 represents the U-Net architecture.

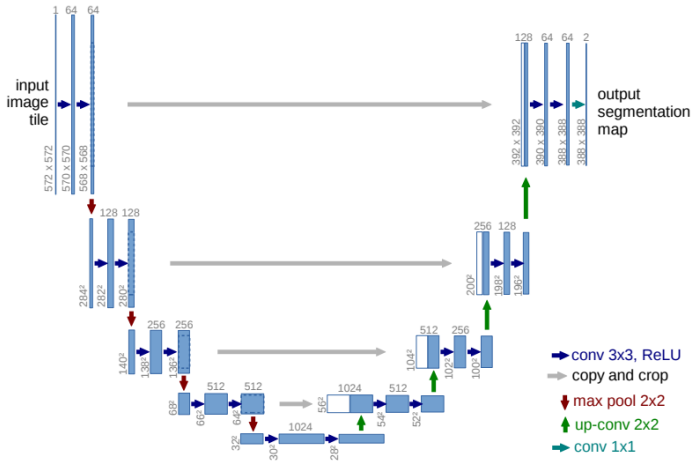


Figure 3.7: U-Net architecture [92].

Man et al. [93] explored the potential of the U-Net architecture for the segmentation of retinal layers in OCT images. The segmentation was carried out for nine boundaries, thus eight retinal layers (ILM, NFL, GCL, IPL, INL, ELM, IS/OS complex, RPE, and BM). The implemented model was used to classify each pixel and it is identical to the U-Net proposed by Ronneberger [92].

After performance evaluation, the segmentation results were found to have, on average, a boundary mean absolute error of 1.144 pixels.

ReLayNet

The ReLayNet architecture was proposed by Roy et al. [94] to address the segmentation of the retina. The authors proposed this new framework to segment the retina into eight layers (ILM, NFL-IFL, INL, OPL, ONL-Inner Segment Myeloid (ONL-ISM), ISE, and

OS-RPE).

Regarding RelayNet (Figure 3.9), the developed algorithm has an encoder-decoder architecture with skip connections, as it is inspired by the U-Net and the DeconvNet [95]. However, unpooling layers were introduced on the decoder, as previous work had shown that they optimize the spatial consistency of the segmentation [95]. Unpooling is a CNN operation used in parallel with the pooling process. The goal of the pooling operation is to reduce the spatial dimensions of the input tensor by retaining the maximum (max-pooling) or average (average pooling) values within a specific neighborhood. The purpose of the unpooling operation is to reverse the process and restore the spatial information lost during the pooling. The authors used max-pooling layers on the encoder as to perform pooling. During this operation, the location of the retained maximum values was stored. Then, when unpooling was applied on the decoder, the current values were restored to their original locations before the pooling operation, while the rest of the values were set to zero. Figure 3.8 shows a diagram that illustrates an example of the max-pooling and unpooling process.

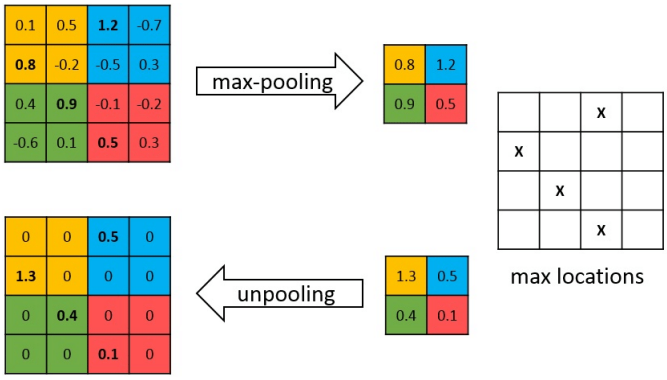


Figure 3.8: Max-pooling and unpooling operations [96].

Besides unpooling, batch normalization layers were also added. Batch normalization is a technique commonly used in deep neural networks to address the issue of internal covariate shift [97], which is related to the unavoidable shift in the distribution of layer inputs as the network parameters are updated. During the training process, the model undergoes layer-by-layer backward updates while assuming that the weights in the layers preceding the current layer are fixed. However, in reality, all layers are modified during

an update, and this update process constantly tries to catch up with a moving target.

The batch normalization layers scale the output of each layer, standardizing the activations of every input variable across the network, enforcing a Gaussian distribution for the inputs, with a mean that approximates zero and a standard deviation that approximates one.

Consequently, the assumptions made by subsequent layers regarding the distribution of inputs during weight updates will remain relatively consistent, experiencing only minor changes. Extensive research [97] has demonstrated that batch normalization substantially stabilizes and accelerates the training process while enhancing the network's robustness, particularly when dealing with poorly initialized weights.

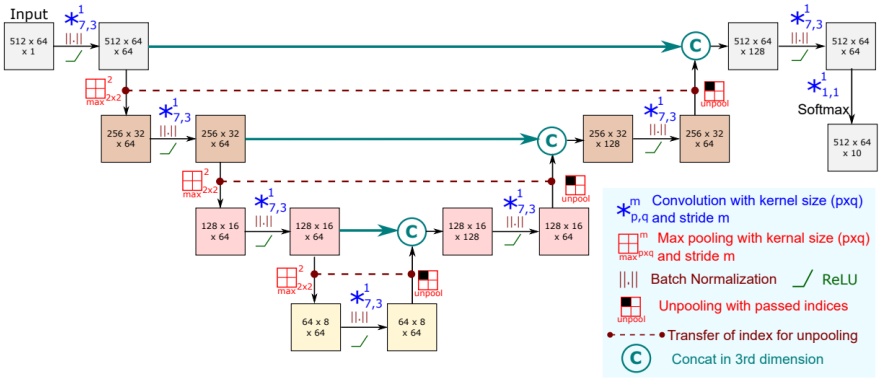


Figure 3.9: RelayNet architecture [94].

The entire eight-layer segmentation procedure was carried out using the algorithm described for semantic segmentation of the different layers, and there was no processing before or after segmentation.

The network was tested, and its performance was evaluated according to the estimation of the thickness of the nine spatial zones of the retina. ReLayNet showed exceptional performance in retinal segmentation with thickness errors in identifying the regions between 0.151 and 0.34 pixels.

Finally, Table 3.1 shows the different networks and their performances.

Table 3.1: Summary of the mentioned approaches, comparing the OCT systems, the error ranges, and the type of data used.

Network	**Reference**	**Error** (px)	**System**	**Data Type**
CNN-GS	Fang et al.[55]	1.260	Spectralis HRA + OCT device	human, age-related macular degeneration
ResNet	Liu et al.[85]	1.215	Spectralis OCT	human, diabetic macular edema
DenseNet	Pekala et al.[89]	1.060	Spectralis OCT	human, non-proliferative diabetic retinopathy
U-Net	Man et al.[93]	1.144	Spectralis OCT	human, normal
RelayNet	Roy et al.[94]	0.151 to 0.340	Spectralis OCT	human, diabetic macular edema

Even though the different algorithms were not used to segment the same layers, the segmentation potential of the U-Net and its derivative RelayNet is clear. Indeed, U-Net inspired networks have been increasingly used to address the retinal layer segmentation problem, using concepts such as residual and dense blocks to supplement the base U-Net architecture. [98, 93]. The same thought process was implemented when developing this study's algorithms, which will be further explored in Chapter 4.

It is also worth noticing that, while the DenseNet may demonstrate a slightly better performance than the ResNet, dense blocks use concatenation, which makes the DenseNet more memory-intensive than the ResNet. Therefore, using it when dealing with smaller data sets is better. Besides this, due to their vast use of connections, besides raising the number of parameters and the computational burden, they are also more prone to overfitting. The direct access of the feature maps to all previous layers may lead to overfitting if the network starts to rely too heavily on redundant or noisy features. Even though residual blocks have a similar concept, in DenseNets, these direct connections are much more frequent along the architecture.

Chapter 4

Materials and Methods

The project aims to develop a system for the automatic segmentation of retinal layers in OCT B-scans using a convolutional neural network. To achieve this goal, the research was mainly focused on the architecture of the proposed network and its flexibility and ability to correctly segment the retina when fed new data.

In this chapter, the materials used, along with the methodologies and procedures implemented, are presented. Additionally, it will cover the different stages of network development and provide a comprehensive description of the model and all the implicit essential components, which include the loss function, type of layers, and hyperparameters.

4.1 Problem Definition and Workflow

The segmentation of any image involves dividing it into meaningful structures or regions. Consequently, this process can be thought of as the classification of each individual pixel into one of these regions (pixel-level classification). The proposed CNN should receive as inputs OCT B-scans, of a predefined size, classify each pixel into one of the 10 classes

(Vitreum, RNFL-GCL, IPL, INL, OPL, ONL, ILS, OLS, RPE, and Choroid), and output a stack of 10 images of the same size of the input one. The output images' pixel values will represent the probability of each pixel belonging to the class. The problem can be formulated as a semantic segmentation task since each pixel will be assigned a class label.

An appropriate approach to this image classification problem is usually training the developed algorithm based on the dataset with pre-labeled images (supervised learning).

Once the problem of image segmentation was defined and framed, a typical implementation workflow of the deep learning model was followed, which includes the following general outline of steps:

- **Data Pre-processing:** Gathering the dataset composed of the OCT B-scans, labeled and unlabeled examples. Pre-processing steps will be performed on these data;
- **Model Selection:** Selection of an appropriate deep-learning model architecture for the segmentation task at hand;
- **Model Training:** Training the selected model, by feeding it the training set (labeled data). The model's progress is monitored, and overfitting is prevented by validating it using a validation set;
- **Hyperparameter Tuning:** Adjustment, by trial and error, of the multiple hyperparameters such as learning rate, batch size, and epochs to optimize the model's performance;
- **Post-Processing:** Application of post-processing techniques to refine and enhance achieved results;
- **Model Evaluation:** Evaluation of the trained model by making predictions on a separate test set of images to assess its performance through several metrics;
- **Fine-tuning and Optimization:** Fine-tuning the model if necessary by incorporating additional techniques such as transfer learning. Optimization of the model's

performance by experimenting with different optimization algorithms and regularization methods.

4.2 Materials

4.2.1 *Hardware*

Regarding the materials used throughout the study, two computers were used to develop and implement the mentioned methodologies and perform the experiments. The platforms had the following specifications:

Computer 1:

- **Operative System:** Linux Ubuntu 20.04.3 LTS;
- **Processor**: 2× Intel Xeon(R) CPU X5660 2.80GHz (24 cores in total);
- **RAM:** 94.4 GiB;
- **Graphics:** NVIDIA Corporation TU104 [GeForce RTX 2080 SUPER 8 GB].

Computer 2:

- **Operative System:** Linux Ubuntu 22.04.2 LTS;
- **Processor:** AMD Ryzen 9 3900X 12-Core Processor x 12;
- **RAM:** 64.0 GiB;
- **Graphics:** NVIDIA Corporation GA106 [GeForce RTX 3060 12 GB].

4.2.2 *Software*

Python

Python is a popular high-level programming language that finds applications in computer vision, data analysis, machine learning, and many other related fields. Known for its simplicity, readability, and intuitive syntax, Python's versatility extends to different platforms, including Windows, macOS, and Linux. In addition, as an open-source language, it offers a comprehensive standard library and a vast collection of third-party packages readily available online. This rich ecosystem and extensive documentation provide developers with a wide range of powerful tools and functionalities to tackle diverse programming tasks effectively.

For these reasons, throughout the development of this research, all neural networks were built, trained, and evaluated using Python 3.10 on PyCharm Professional Edition, an integrated development environment (IDE) used for programming in Python.

Keras

Compatible with Python 3.5 or higher, Keras is a high-level deep learning library tightly integrated with TensorFlow's deep learning framework that supports convolutional and recurrent networks. With Keras, it's possible to quickly design and train complex neural networks by stacking layers and configuring their parameters through a wide range of built-in layers, activation functions, loss functions, and optimizers.

Keras excels in its straightforwardness, modularity, and easy extensibility, allowing for efficient experimentation and iterative model improvement. Owing to this, it was decided to work on top of the Tensorflow graph flow processing library.

4.2.3 Data

The data used for this study was acquired in the *Coimbra Institute for Clinical and Biomedical Research (iCBR), Faculty of Medicine, University of Coimbra* and the *Coimbra Institute for Biomedical Imaging and Translational Research (CIBIT)*, using two different systems: the Phoenix OCT and the Bioptigen.

Phoenix OCT

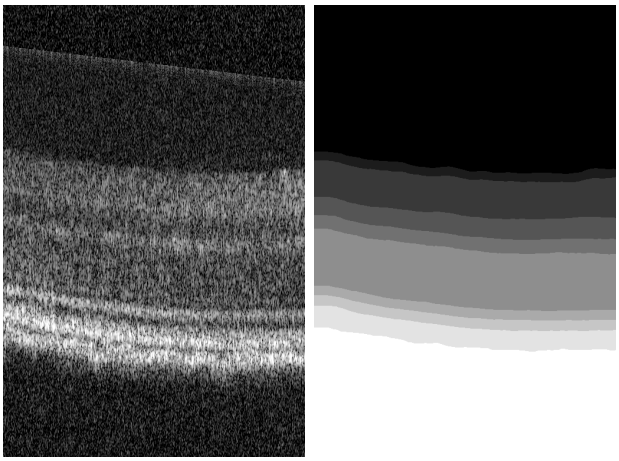
Mice retinas were imaged using the Micron IV OCT System (Phoenix Technology Group, Pleasanton, CA, USA) to capture 1024×512 pixels B-scans, saved as non-compressed TIFF files. The imaging system offers a depth scanning of up to 1.4 mm and a depth resolution of 3 μm. These characteristics are determined by the bandwidth and central wavelength of the superluminescent diode, which are 160 nm and 830 nm, respectively.

All scans were captured in the same retinal region by the same operator. The optic disc was utilized as a reference point for alignment, ensuring that scans were horizontally centered and positioned vertically above it.

This OCT system was used to collect the main dataset (dataset 1 - DS1) for training the network, which comprised image data from 57 triple transgenic animal models of Alzheimer's disease (3×Tg-AD) and 57 wild-type (WT) mice. The data consisted of 3000 labeled OCT B-scans acquired for both eyes, at 1, 2, 3, and 4 months of age. Labeling was accomplished through a semi-automatic process, where an expert iteratively corrected the systems segmentation over time and layers were aggregated to create a manual segmentation. This dataset was divided into training, validation, and testing as explained in the next section (Section 4.3). In Figure 4.1, an example B-scan from dataset DS1 is shown, along with its label.

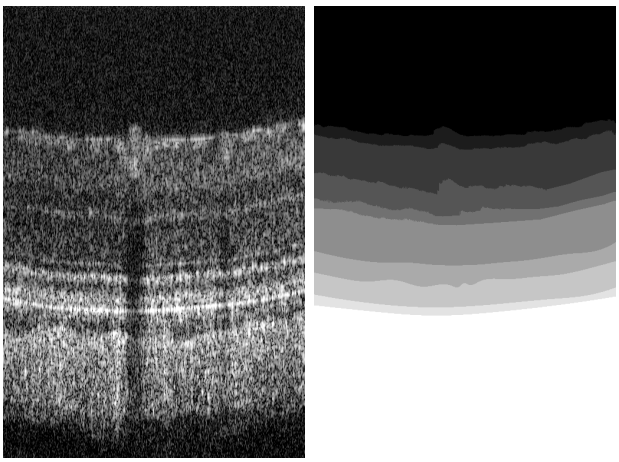
Besides DS1, additional OCT B-scans collected with the same acquisition system and method were provided (dataset 2 - DS2), containing samples from both eyes of rats from WT and type 1 diabetes, at 1, 2, and 4 weeks after diabetes onset. Type-1 diabetes was induced by a single intraperitoneal injection of streptozotocin (STZ; 65

mg/kg; Sigma-Aldrich, Merck KGaA, Darmstadt, Germany). The same labeling process used for DS1 was also used to obtain these labels. These images included 21 labeled B-scans and they were only used for testing. In Figure 4.2, a B-scan from dataset DS2 and its label are displayed.



(a) (b)

Figure 4.1: An OCT B-scan example (a) and the corresponding manual segmentation (b), from dataset 1 - DS1.



(a) (b)

Figure 4.2: An OCT B-scan example (a) and the corresponding manual segmentation (b), from dataset 2 - DS2.

Bioptigen

Aiming to test the proposed solution on data gathered by a distinct acquisition system, additional B-scans, collected by a Bioptigen OCT system were gathered (dataset 3 - DS3). This imaging system captures 1024×512 pixels B-scans and provides a depth resolution of 4.5 μm.

Recalling that the final requisite consisted of applying transfer learning methods to data from the Bioptigen system, a manual segmentation process was required to create the target images needed. Using the GIMP application, 60 B-scans were randomly selected from 3 mice and manually segmented, 30 from the left eye and 30 from the right eye, to prompt variability. Ideally, a larger quantity of labeled data would be preferable, but manual segmentation requires significant time and effort, and the process is slow and resource-intensive. The 60 B-scans and their associated masks were split into training and validation sets as explained in the next section (Section 4.3). All segmentations were verified before the network underwent training by Professor Doctor Rui Bernardes. For testing, another 109 B-scans were selected randomly from several Bioptigen's volumes, corresponding to 5 mice, 62 from the left eye and 47 from the right eye. Below is a B-scan belonging to DS3 (Figure 4.3).

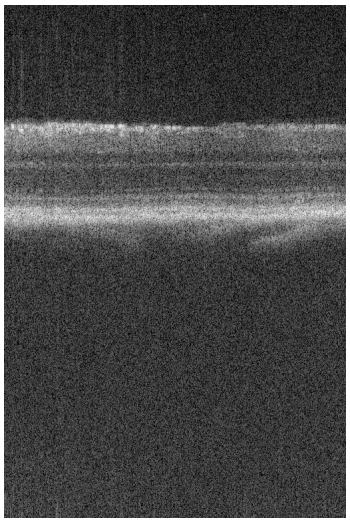


Figure 4.3: An OCT B-scan example from dataset 3 - DS3.

4.3 Pre-Processing

As mentioned above, both the DS1 dataset and the DS3 dataset had to be divided into subsets for the two different training phases, the main training procedure, and the transfer learning.

Dataset 1, containing 3000 labeled B-scans, was randomly divided into training, validation, and test groups per animal in a proportion of 80/20/20, meaning that a first split into training+validation and test sets was performed with a ratio of 80 and 20%, respectively, followed by a second split of the former into training and validation sets also with a ratio of 80 and 20%, respectively.

As for dataset 3, the 60 labeled B-scans were split into training and validation sets, 80 and 20% ratio, respectively.

All data splits are performed on the animal IDs and the B-scans from each animal are assigned to the resulting set. Thus, all B-scans from a given animal are part of either the train, test, or validation sets.

Normalization was applied to all OCT images. Normalizing the dataset involves transforming the pixel range of the image data to a new standardized scale. The goal is to bring all features or variables to a similar range.

Normalizing inputs ensures that the pixel values have similar scales and distributions, thus mitigating possible biases due to different pixel value ranges in the images, ensuring that each input image contributes equally to the learning process. It also improves the speed of convergence during training, potentially optimizing the efficiency and performance of the model. In addition, it reduces the variation between images and becomes easier to compare and interpret the data.

Two of the most commonly employed techniques for this procedure are z-score normalization and min-max normalization. Min-max normalization involves scaling the values according to the minimum and maximum values of the sample. Herein, z-score normalization was used to standardize the image data. For each image in the dataset, the mean, μ, and standard deviation, σ, were calculated, and then the following equation was applied to every image pixel:

$$Z = \frac{x - \mu}{\sigma} \Rightarrow img_{standardized} = \frac{img - \bar{img}}{s_{img}} \ , \tag{4.1}$$

where x is the original data point, μ is the mean of all values in the image, σ is the standard deviation, and z is the standardized value for x.

Simultaneously, recalling the multi-label classification task at hand, the output classes resulting from the model classification will be eight retinal layers plus the Vitreum and Choroid regions, which means there will be 10 different classes.

To handle this kind of categorical labels, it's necessary to extract each layer separately from the manually segmented image, which together will compose the target images for the machine learning algorithm.

Consequently, the next stage consisted of performing one-hot-encoding on each manually segmented image, resulting in a stack of 10 images, one for each class, as shown in Figure 4.4.

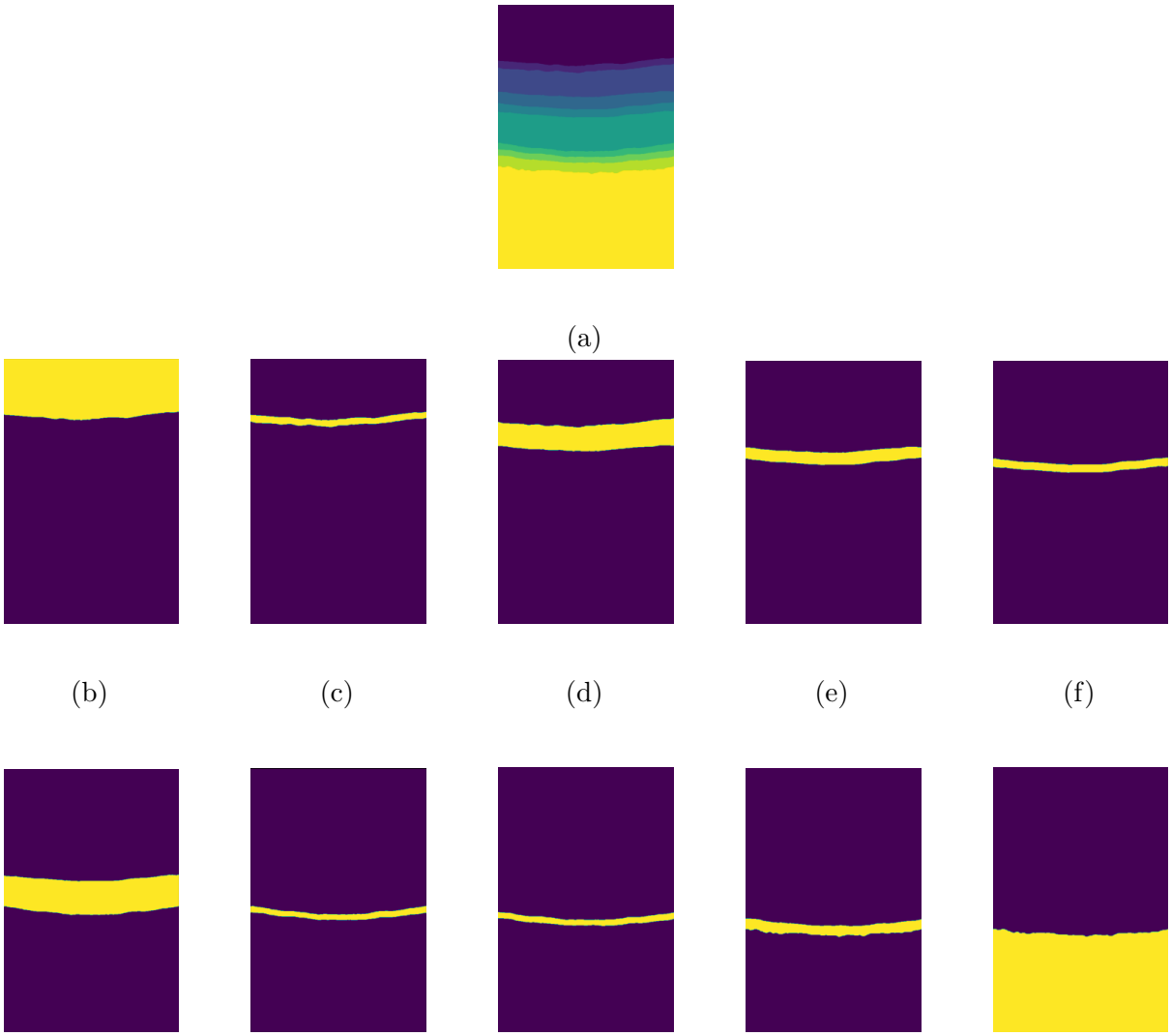


Figure 4.4: Manually segmented image (a) from DS1 and each layer image after the one-hot-encoding process implementation: (b), (c), (d), (e), (f), (g), (h).

Finally, all training and validation data underwent data augmentation, prior to each training stage. This technique was applied to ensure that the developed network had access to a substantial amount of diverse data.

Therefore, it aimed to mitigate overfitting and enhance the model's ability to make accurate predictions on new images, ultimately improving its generalization capabilities.

Hence, the following geometric transformations were randomly applied:

- **Rotations:** Consists of randomly rotating the original image according to a desired angle. Rotations of up to 15° were applied;
- **Reflections (Flipping):** Involves randomly flipping images along an axis of symmetry horizontally or vertically. Only horizontal flips were performed.

In these transformations, padding is applied according to the value of the nearest pixel. These transformations have been found to significantly enhance deep learning algorithms' performances while keeping a reasonably fast computational speed (convergence velocity) [99, 100].

Data Resizing

The goal of this work envisions a flexible approach for multiple animal models and systems. Input size is different between systems and the proposed approach must account for it. The standard input size for the network was set based on the main dataset as 768×512 pixels. Thus, a protocol was then defined to pad/crop irregular inputs before pre-processing.

To accomplish this, a function was created to crop input images according to their respective regions of interest and specific output size.

Initially, the function reads the image in grayscale mode and ensures pixel values are rescaled within the range of 0 to 255. Then, it sets a minimum intensity threshold for the region of interest (ROI) at 50, to facilitate its determination. The threshold value was empirically set at 50, but its applicability to different datasets is not invalidated, as all images are initially set between 0 and 255, and this process is applied before the B-scans are standardized. The rationale of this approach is the high reflectivity found in the RPE. Utilizing the Opencv library, the perimeter of the ROI is identified, and its center is calculated. Finally, the procedure resorts to the PIL library to crop the original image, ensuring the crop centers around the calculated center while spanning specific dimensions—a width of 512 pixels and a height of 768 pixels.

4.4 Model Architecture

Two main architectures were implemented: an ***Attention-Res-U-Net*** and a teacher-student generative adversarial network, ***teacher-student GAN***. The thought process behind these choices will be explained throughout this section. All the layers that composed each framework were defined using *Keras*.

4.4.1 *Attention-Res-U-Net*

The ***Attention-Res-U-Net*** or ***Att-Res-U-Net*** is a *U-Net* based architecture.

U-Net

The U-Net architecture is widely used for semantic segmentation. This network is composed of double convolution blocks with the ReLU activation function, as shown in Figure 4.5.

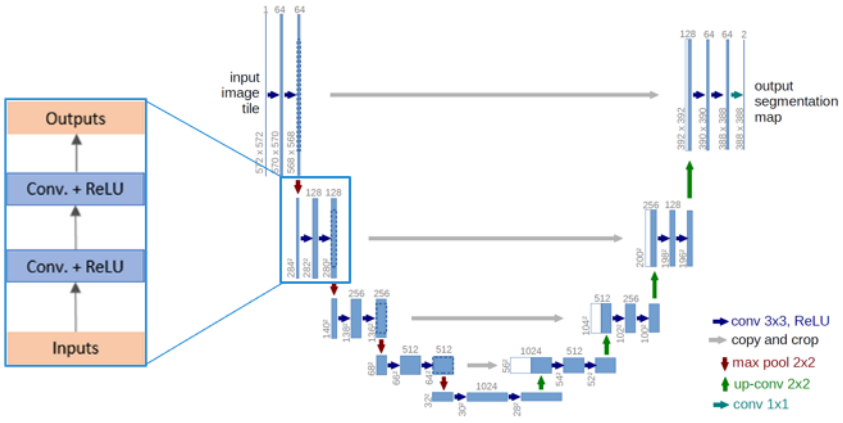


Figure 4.5: U-Net architecture [92] (adapted scheme).

Accordingly, for both the down-sampling path and the up-sampling path, 3×3 convolution layers with a stride of 1 were used, followed by ReLU units. At the end of each

encoder block, 2×2 max-pooling layers were also applied, with a 2×2 kernel and a stride of 2. Each encoder block is followed by a skip connection that concatenates the block's output feature maps to the respective decoder block. The decoder culminates with a final 1×1 convolution layer with a Softmax activation function since this task involves multi-class segmentation, to match the number of channels to the number of classes.

Due to memory resources, eight down-sampling blocks and eight up-sampling blocks were used. Additionally, a batch normalization layer is placed after every convolution layer, before the activation function. As for regularization, a dropout layer was applied at the end of the network, just before the last convolution layer, with a dropout probability of 0.3.

Res-U-Net

Effectively, the segmentation requires a deep model given that it needs to be able to learn all the different structures the retina layers may present. However, with a deeper U-Net, the gradients along the layers decrease during backpropagation. The values of the gradients may reach such small values (near zero) that the updates on the weights will be almost insignificant (vanishing gradient). Thus, the learning process of the model becomes very hard and slow (low convergence speed).

Facing this issue, the solution is substituting the standard U-Net blocks with residual convolution blocks [84], which leads us to a ***Res-U-Net*** framework.

The shortcuts for the identity function x introduced by the residual blocks, as shown in Figure 3.2, allow the propagation of gradients to the initial layers, which mitigates the vanishing gradient issue and the degradation of the network's performance. In addition, the forward propagation velocity also increases.

Attention-Res-U-Net

Finally, an attention mechanism was added, originating the definitive ***Attention-Res-U-Net***.

Looking back at the skip connections along the *U-Net* framework, they concatenate the encoder's feature maps with the corresponding decoder layers, making it possible to

retain good spatial information along the network.

However, the received featured maps in the decoder won't have a feature representation as significant and useful as the one from objects in deeper layers. As the data is processed through the multiple layers, the model extracts increasingly complex features but it also becomes more difficult for it to identify the most relevant information.

The proposed attention mechanism (Figure 4.6) helps the model focus on the most relevant parts of the image, by assigning weights to individual pixels according to their relevance. During the training process, higher weights are allocated to pixels belonging to significant regions, while lower weights are assigned to pixels in less important areas. The significance of each pixel, thus region, is learned throughout the training process and can be increased or decreased. As a result, the model progressively learns to selectively "pay attention" to specific parts of the input, ideally, the target regions, when making predictions.

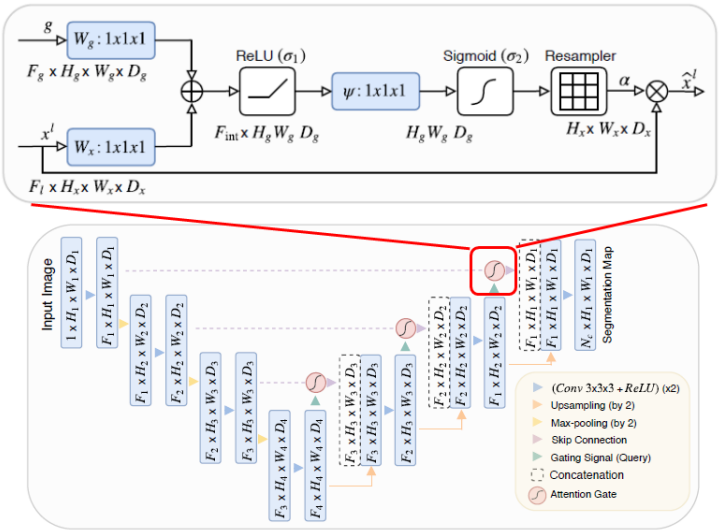


Figure 4.6: Attention U-Net (bottom) and attention mechanism process (top) [101].

Two inputs are received at the attention gate: the gating signal, g, from the previous layer and x from the skip connection, where g will have a better feature representation (deeper layer) and x better spatial information (earlier layer). The product of x with the calculated weights is returned, taking into account the g map, and forwarded to the next layer.

Implementing this attention mechanism, combining both components g and x in the

skip connections, allows a flow of spatial information along the network with a more significant and localized feature representation.

A detailed scheme of the architecture can be found in Appendix A.

Loss Function

A categorical cross-entropy (CE) loss was used. This function is widely used in segmentation tasks with multiple classes, such is the case.

However, the dataset is unbalanced, given that the retinal layers have different thicknesses. Therefore, to improve performance in underrepresented layers or classes, it's necessary to define weights for each one. These weights were calculated to be inversely proportional to class frequency.

A weighted categorical cross-entropy (WCE) loss was then implemented, which is basically the former categorical cross-entropy loss function pondered with class weights. It can be defined by the following expression, where w_i is the weight associated with a class (layer):

$$WCE = -\sum_{i}^{m} y_i \times log(\hat{y}_i) \times w_i \tag{4.2}$$

However, this loss function was still not sufficiently optimized to allow the correct adjustment to the layer boundaries. Even though the layer segmentation may be valid, the boundaries obtained were flat and inaccurate. The main issue is that the critical regions are when two classes border one another. Nevertheless, the WCE loss function weights every pixel of a given class the same. A loss function that considered the pixel distance to these boundaries was needed to emphasize and enhance the influence of pixels located in regions closer to them.

To obtain this boundary distance component, the function *euclidean distance transform* was built to calculate the inverse of the Euclidean distance transform and output an image that contains pixel weights (from 0 to 1). The pixels closest to the boundaries have the largest weights and as the pixels get further from the edges, their weights get progressively smaller. Once the distance from the boundary exceeds a threshold distance of 5 pixels,

the weights become constant at a value w_{min}.

This new map of weights, d_{true}, was multiplied point-by-point by the already-defined loss that considered the layer (class) weights:

$$loss = -\sum_{i}^{m} y_i \times log(\hat{y}_i) \times w_i \times d_{true} \tag{4.3}$$

Finally, a new term $(1 - p_t)^\gamma$ was introduced, where p_t is the probability of a pixel belonging to a class predicted by the model ($\hat{y}_i$) and γ an adjustable focus parameter:

$$weighted\ focal\ loss \quad = -\sum_{i}^{m} y_i \times log(\hat{y}_i) \times w_i \times d_{true} \times (1 - \hat{y}_i)^\gamma \tag{4.4}$$

This term is characteristic of the focal loss [102], which is an extension of the cross-entropy loss, and its purpose is to focus the model on hard-to-classify examples. The loss for well-classified samples, when the model predicts the correct class with $p_t > 0.5$, is reduced, and the loss for hard-to-classify samples, when the model predicts the correct class with $p_t < 0.5$, is increased, as shown in the graph below (Figure 4.7). The γ parameter was adjusted manually through several experiments, and it was found that $\gamma = 2.0$ worked the best.

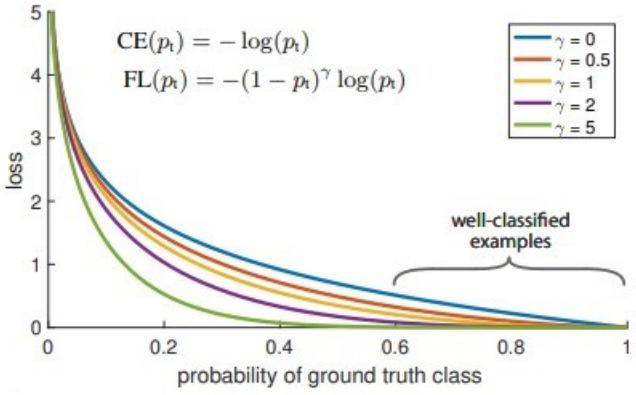


Figure 4.7: Focal loss vs. predicted model probability function for a class [103].

Consequently, it not only reduces the class imbalance, improving the minority classes' (thinner layers) performance but also enhances the performance relative to the classification of hard-to-classify pixels inside each class.

Hyperparameters

Besides the topological structure of the network and the loss function, additional hyperparameters had to be defined when projecting this CNN.

After several adjustments by trial and error, manually, some essential hyperparameters were specified as presented in table 4.1, as they were found to work best throughout the experiments.

Hyperparameter	**Value**
Dropout	0.3
Optimizer	Adam
Learning Rate	0.0001
Batch Size	4
Epochs	250
Steps per Epoch	$\frac{n_{images}}{batchsize}$

Table 4.1: Defined hyperparameters for training the Attention-Res-U-Net model.

4.4.2 *Teacher-Student GAN*

A generative adversarial network (GAN) was also put into practice. Recently, the potential of GANs has been in evidence [90, 104].

This type of architecture is formed by two sub-models (Figure 4.8): the generator (*student*), which generates the target images, and the discriminator (*teacher*), which classifies them as true images or generated ones. As the name indicates these models are adversarial, i.e., their objectives oppose. The generator is trying to 'fool' the discriminator while the discriminator attempts to 'punish' the generator.

Given that the GAN frameworks can be used as a form of data augmentation, this is also one of the appeals of these networks: the production of more training data than the provided input. GANs can generate realistic samples and increasingly improve the data quality by comparing it to similar data and making corrections.

The following logic scheme translates the projected network's information flow:

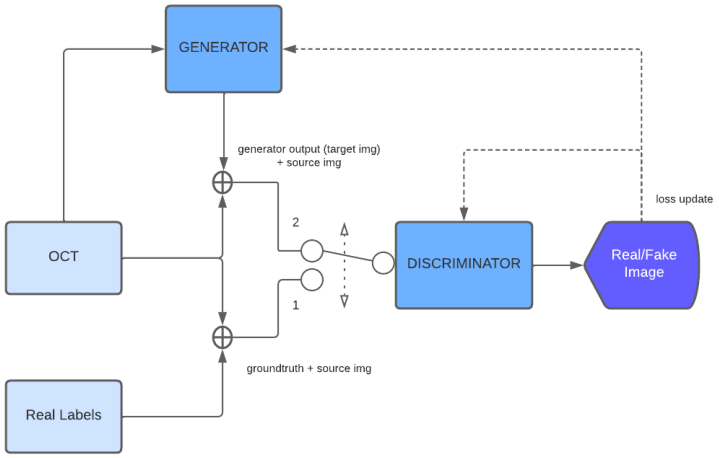


Figure 4.8: Diagram of the information flow of the proposed GAN algorithm.

For the most part, this network's architecture was based on the Pix2Pix GAN proposed in [105]. For the generator, the Attention-Res-U-Net framework was used, while for the discriminator, a more straightforward CNN architecture was used: 5 blocks of 4×4 convolution layers with a stride of 2, followed by batch normalization layers (except in the first block) and LeakyReLU units, with an alpha of 0.2. The final layers are a 4×4 convolution layer, followed by a Sigmoid activation unit.

A particularity of the discriminator relates to the fact that the probability is evaluated on an image region or patch instead of the image as a whole, in this case, a 48×32 sized patch, since the last convolution layer has a 4×4 kernel size and the images are of size 768×512 pixels. This is called a PatchGAN model. A scheme of the PatchGAN model can be found in Appendix A.

Loss Function

Regarding the choice of loss functions, distinct functions were employed for each of the two models. Specifically, for the generator the preferred loss function was the weighted categorical loss, due to memory constraints. For the discriminator, a binary cross-entropy loss was used. This loss quantifies the disparity between predicted probabilities and labels due to its straightforwardness and capacity to adeptly direct the discriminator's learning process.

Hyperparameters

Discriminator and generator parameters were adjusted similarly to the ones from the standalone Attention-Res-U-Net and are summarized in Table 4.2.

Table 4.2: Defined hyperparameters for training the Teacher-Student model.

	Attention-Res-U-Net	**PatchGAN**
Dropout	0.3	-
Optimizer	Adam	Adam
Learning Rate	0.0001	0.001
Batch Size	4	
Epochs	100	
Steps per Epoch	$\frac{n_{images}}{batchsize}$	

GANs tend to not converge or converge very slowly. Furthermore, the discriminator may become too successful in its task, leading to gradient vanishing problems. For these reasons, parameterization should be carefully considered. Hence, mismatched learning rates between the generator and discriminator help strike a balance in the training process. The generator benefits from a lower learning rate because this allows it to make gradual updates and converge slowly toward generating realistic samples. On the other hand, a faster learning discriminator (higher learning rate) ensures a dynamic and competitive training process. This speed prevents the discriminator from failing to provide the feedback needed for effective generator training, thereby continuously challenging the generator to improve its sample generation [106].

Adjusting the learning rates was done empirically through experimentation, and resulted in 0.0001 for the Attention-Res-U-Net (generator) and 0.001 for the PatchGAN (discriminator).

Network Training

After the architectures, loss functions, and hyperparameters were defined, the network training phase succeeded. It was performed on both the Attention-Res-U-Net and the Teacher-Student GAN. Results from this stage can be found in the next chapter. During training, accuracy and loss were monitored for the Attention-Res-U-Net, while for the

Teacher-Student GAN loss was monitored along with visual evaluation of the quality of the generated images. Periodically throughout the training procedure, the student model would make some predictions and display them. This form of monitoring was implemented to know when to stop the training process.

Overall, two models were trained and saved using DS1, one for each network, and a third one was also trained during the transfer learning step for DS3, using the already-trained Attention-Res-U-Net model.

This last training phase consisted of the application of a transfer learning method, where a new model using the Attention-Res-U-Net architecture was trained. The initial weights for this new model were set as the concluding weights obtained from the previously trained Attention-Res-U-Net model.

It's also worth noting that besides changing the number of training epochs to 50, the Adam optimizer was set with a learning rate of 0.001 for this training.

4.5 Post-Processing

After generating the predicted segmentations (probability maps) from the network, post-processing was applied to refine and enhance the obtained results and produce the final interfaces that limit each layer.

This can involve a wide range of techniques and algorithms to improve the visual quality, correct imperfections, enhance details, or transform the image in some way.

It is worth noting that, since the network outputs 10 images of the 10 classes, the same interface will be predicted in more than one image (Ex: In the 2nd layer image, the upper boundary will coincide with the lower boundary of the 1st layer image.). Therefore, all pairs of interface predictions were considered and averaged. To do this, an image of all the layers summed up below the interface and an image of all the layers summed up above the interface were generated, as depicted in Figure 4.9, for all interfaces. For the sake of simplicity, the post-processing steps will refer to the boundary extraction per image.

(a) (b)

Figure 4.9: Depiction of an example of each pair of images generated for boundary detection.

The method to extract the layer interfaces consisted of the following steps:

- **Interface Refinement and Detection:** Interface identification on the image, with previous manipulation to make them sharper and easier to detect.

  First, to use the library destined for this procedure, OpenCV, each image had to be scaled from [0, 1] to [0, 255]. Initially, to slightly enhance the boundary, a binary threshold is set, where pixels above the threshold are set to white (255) and pixels below the threshold are set to black (0). The threshold was determined empirically to be 255/2.5, but this does not compromise the application of this method to other types of data, since all images have the same range, at this point, and variations of the threshold do little to no difference in the final result.

  Afterward, to make sure small artifacts or noise were removed from the image, some basic morphological operations of erosion and dilation were employed. Erosion allows the corrosion of a region's boundaries, considering a kernel with a certain connectivity (number of neighboring pixels). Each pixel of the original binary image is considered 1 if every pixel in its connectivity is also 1. Dilation does the exact reverse, broadening the region's boundaries. Each pixel of the original binary

image is considered 1 if at least one pixel in its connectivity is also 1. This was performed using the function *morphologyEx* with an elliptical-shaped kernel of 7×7. An example of a resulting image from this refinement process can be shown below (Figure 4.10).

Figure 4.10: Final product generated from the refinement of the boundary image.

Finally, the detection of the boundaries is performed using the Canny edge detection algorithm, through the use of *cv2.Canny*.

- **Boundary Coordinates Filtering**: Location of boundary pixel coordinates and screening of coordinates to obtain a single vector of x and y values.

  Following the interface detection process, a final step is required to ensure that the resulting interface consists of a single set of x values, each uniquely paired with a single corresponding y value. This step involves locating the boundary pixels' coordinates *(x,y)* and filtering them, by screening the coordinates of the detected edge to eliminate any redundancy or ambiguity of the boundary location.

  Hence, the positions of pixels with an intensity equal to 255 were located and through several loops, the coordinates were screened for possible multiple y coordinates. If there was in fact more than one y value for an x coordinate, an average for all the y values was done, like so:

$$(x, y)_{final} = (x, \bar{y}) = (x, \frac{\hat{y}_1 + \hat{y}_2 + ... + \hat{y}_n}{n}) \tag{4.5}$$

Thus, the final output after applying this post-processing stage to the network's predictions is nine sets of coordinates for each layer interface.

Ultimately, these coordinates were drawn onto the corresponding test B-scan, resulting in images of 768×512 pixels with the delineated layer interfaces. An example of a resulting image with the interfaces can be observed in Figure 4.11.

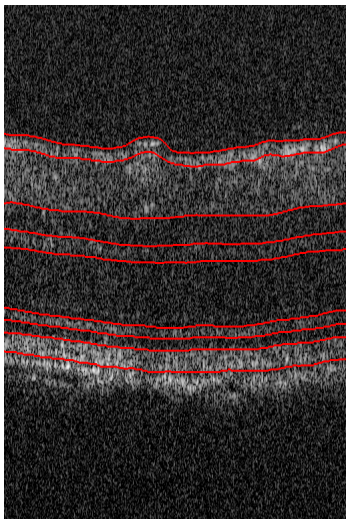


Figure 4.11: B-scan with the delineated layer interfaces generated from the post-processing of the network output.

4.6 Performance Indicators

In the context of semantic segmentation, the evaluation metrics must assess pixel-level classification performance. Thus, to evaluate the developed models, the following metrics were used as performance indicators:

- **Accuracy:** given as the ratio between the number of correctly classified pixels and the total number of pixels:

$$Accuracy = \frac{number\ of\ correctly\ classified\ pixels}{total\ number\ of\ pixels} \tag{4.6}$$

- **Recall (Sensitivity):** given as the ratio between the number of correctly classified

pixels and the total number of pixels in the ground truth that ractually belong to the class:

$$Recall = \frac{number\ of\ correctly\ classified\ pixels}{total\ number\ of\ ground\ truth\ instances\ of\ class} \tag{4.7}$$

- **Precision:** given as the ratio between the number of correctly classified pixels and the total number of pixels that the model predicted as belonging to the class:

$$Precision = \frac{number\ of\ correctly\ classified\ pixels}{total\ number\ of\ predicted\ instances\ of\ class} \tag{4.8}$$

- **F1-score:** given as a harmonic mean between recall and precision. Its values can range from 0 to 1, with 1 being the maximum value of precision:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{4.9}$$

- **Dice Coefficient:** given as twice the area of overlap divided by the total number of pixels in both the ground truth and predicted images. It ranges from 0 to 1, with 0 being no overlap and 1 signifying the greatest similarity between ground truth and predicted (Figure 4.12);

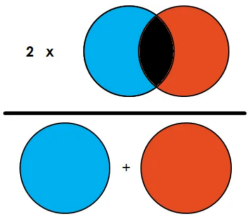


Figure 4.12: Illustration of the dice coefficient [107].

- **IoU (Intersect over Union):** given as the area of overlap between the predicted segmentation and the ground truth divided by the area of union between the predicted segmentation and the ground truth. It ranges from 0 to 1, with 0 being no overlap and 1 signifying perfectly overlapping segmentation (Figure 4.13).

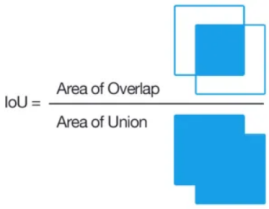


Figure 4.13: Intersect over Union calculation visualized [108].

Model evaluation through these metrics is carried out directly on the network output, before the post-processing, since these metrics are related to the pixel-level classification ability of the algorithm.

After the post-processing stage, the boundary prediction error is evaluated. Thus, the following error distance metrics are also calculated for each interface:

- **Mean Error (ME):** average of the distance between the predicted ($\tilde{y}_i$) and the original/true (y_i) values:

$$ME = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i) \tag{4.10}$$

- **Mean Absolute Error (MAE):** average of the absolute distance between the predicted ($\tilde{y}_i$) and the original/true (y_i) values:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \tilde{y}_i| \tag{4.11}$$

- **Mean Squared Error (MSE):** square average of the distance between the predicted ($\tilde{y}_i$) and true (y_i) values:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2 \tag{4.12}$$

Chapter 5

Results and Discussion

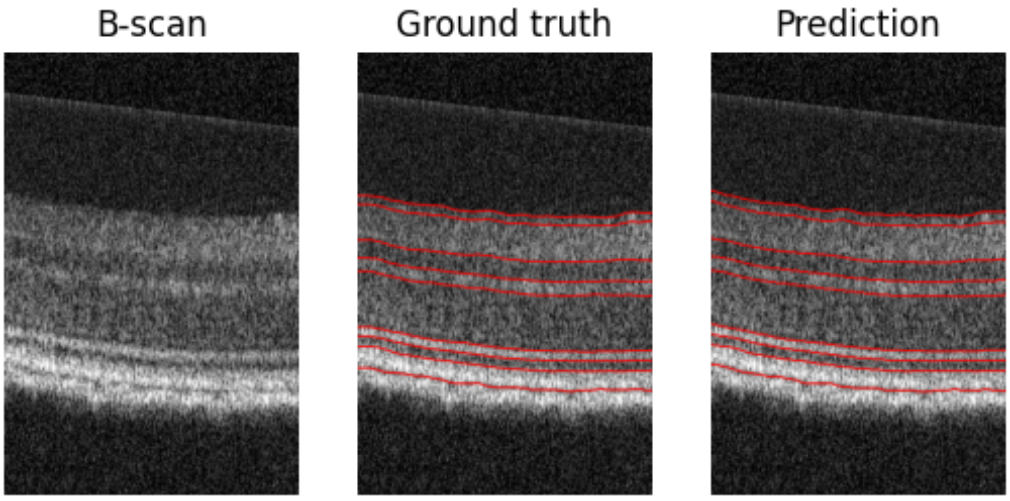
5.1 Phoenix OCT

5.1.1 Dataset 1 - DS1

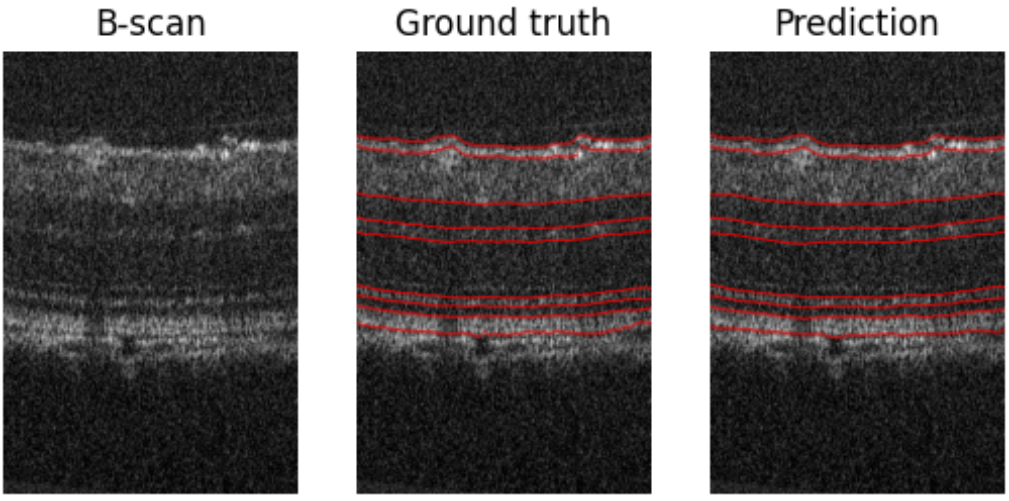
In this subsection, the segmentation results of the DS1 dataset encompassing WT and 3×Tg-AD mice OCT images obtained from the Phoenix OCT system, will be presented and discussed. These results were obtained using the developed networks, Attention-Res-U-Net and Teacher-Student GAN. Additional segmentation results can be found in Appendix A.

Attention-Res-U-Net

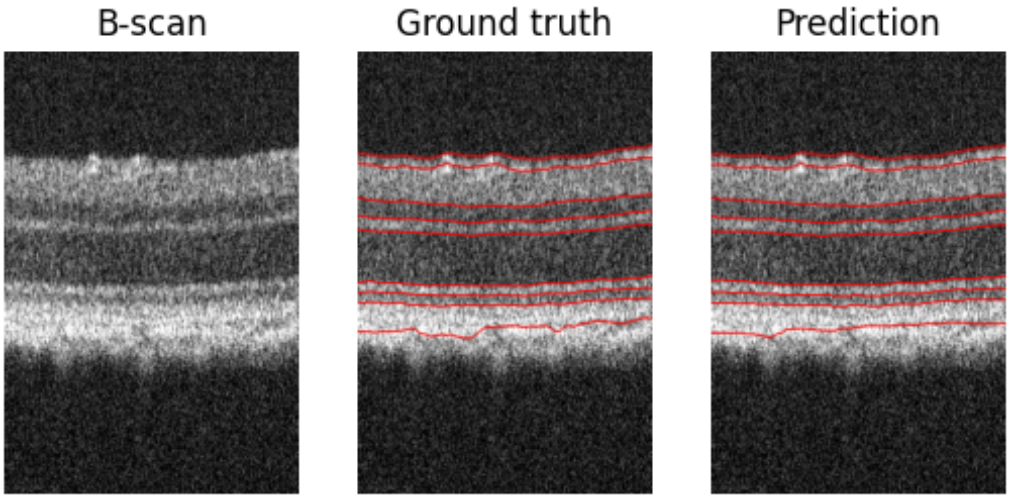
The training duration for this model was 20 hours and 49.8 minutes, spanning 250 epochs. Graphs illustrating the progression of loss and accuracy can be found in Appendix A for both the training and validation stages. As the model learned, the loss decreased while accuracy progressively improved, stabilizing at a value and maintaining that level for the remaining training duration.


B-scan
Ground truth
Prediction

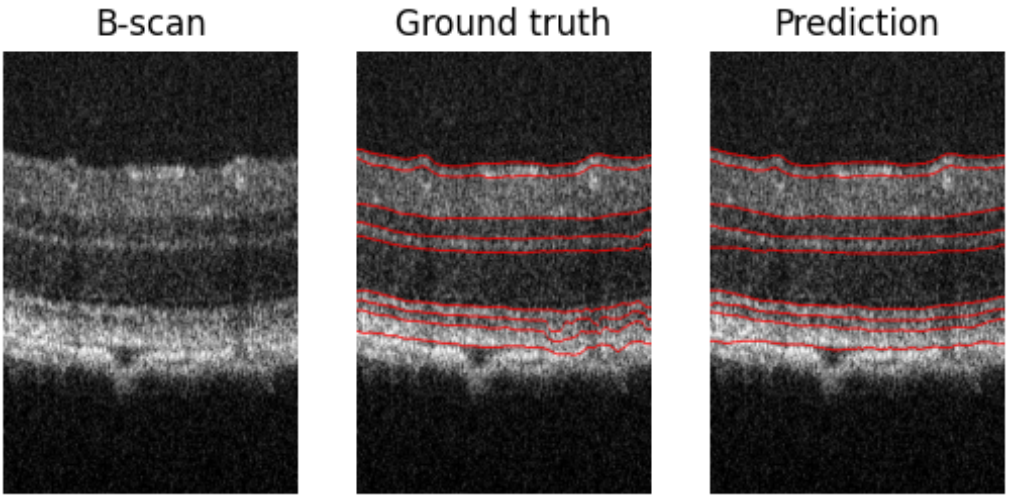
(a)


B-scan
Ground truth
Prediction

(b)



(c)



(d)

Figure 5.1: Selection of predicted layer interfaces by the Attention-Res-U-Net, corresponding ground truth segmentations, and original B-scans for the DS1 test set.

Table 5.1: Performance metrics for the classification into layers of the DS1 test set using the Attention-Res-U-Net model, along with the average and weighted average. The weighted average is pondered using the class weights, w_i.

	F1-Score			**Dice Coefficient**			**Recall**		
	Median	**Q1**	**Q3**	**Median**	**Q1**	**Q3**	**Median**	**Q1**	**Q3**
Vitreous	0.998	0.997	0.998	0.997	0.996	0.998	0.995	0.994	0.997
RNFL-GCL	0.960	0.944	0.969	0.938	0.919	0.947	0.995	0.990	0.998
IPL	0.989	0.985	0.990	0.976	0.970	0.979	0.981	0.974	0.985
INL	0.978	0.972	0.982	0.940	0.928	0.946	0.985	0.979	0.988
OPL	0.969	0.961	0.974	0.883	0.870	0.891	0.991	0.987	0.994
ONL	0.990	0.988	0.992	0.961	0.957	0.963	0.982	0.977	0.985
ILS	0.968	0.963	0.971	0.857	0.847	0.864	0.992	0.989	0.995
OLS	0.975	0.968	0.980	0.888	0.875	0.895	0.985	0.980	0.989
RPE	0.966	0.955	0.973	0.899	0.884	0.908	0.979	0.974	0.984
Choroid	0.996	0.995	0.997	0.991	0.988	0.993	0.993	0.990	0.995
Average	0.979	0.973	0.983	0.933	0.923	0.938	0.988	0.983	0.991
Weighted Average	0.980	0.974	0.983	0.936	0.927	0.941	0.988	0.983	0.991

	Precision			**Accuracy**			**Intersect over Union**		
	Median	**Q1**	**Q3**	**Median**	**Q1**	**Q3**	**Median**	**Q1**	**Q3**
Vitreous	1.000	1.000	1.000	0.995	0.994	0.997	0.994	0.992	0.995
RNFL-GCL	0.930	0.903	0.948	0.995	0.990	0.998	0.883	0.850	0.900
IPL	0.997	0.995	0.998	0.981	0.974	0.985	0.953	0.943	0.958
INL	0.972	0.962	0.979	0.985	0.979	0.988	0.886	0.865	0.898
OPL	0.948	0.936	0.956	0.991	0.987	0.994	0.790	0.770	0.803
ONL	1.000	0.999	1.000	0.982	0.977	0.985	0.925	0.917	0.929
ILS	0.945	0.937	0.951	0.992	0.989	0.995	0.750	0.735	0.760
OLS	0.965	0.957	0.972	0.985	0.980	0.989	0.798	0.778	0.810
RPE	0.955	0.937	0.965	0.979	0.974	0.984	0.816	0.793	0.832
Choroid	1.000	1.000	1.000	0.993	0.99	0.995	0.982	0.977	0.986
Average	0.971	0.963	0.977	0.986	0.983	0.991	0.878	0.862	0.887
Weighted Average	0.973	0.965	0.978	0.986	0.983	0.991	0.883	0.868	0.892

RNFL-GCL: Retinal Nerve Fiber Layer-Ganglion Cell Layer; IPL: Inner Plexiform Layer; INL: Inner Nuclear Layer; OPL: Outer Plexiform Layer; ONL: Outer Nuclear Layer; ILS: Photoreceptor Inner Segments; OLS: Photoreceptor Outer Segments; RPE: Retinal Pigment Epithelium.

Table 5.2: Distance errors per interface for the predicted segmentation by the Attention-Res-U-Net model of the DS1 test set.

	Mean Squared Error (μm)			**Mean Absolute Error** (μm)			**Mean Error** (μm)		
	Median	**Q1**	**Q3**	**Median**	**Q1**	**Q3**	**Median**	**Q1**	**Q3**
Vitreous-RNFL	1.559	1.063	2.354	1.045	0.819	1.312	-1.018	-1.298	-0.778
GCL-IPL	0.988	0.653	1.718	0.676	0.522	0.962	0.295	0.066	0.688
IPL-INL	1.762	1.225	2.604	1.014	0.823	1.260	-0.840	-1.165	-0.498
INL-OPL	0.848	0.688	1.129	0.664	0.583	0.771	-0.219	-0.437	-0.013
OPL-ONL	1.457	1.005	2.464	0.916	0.732	1.219	0.756	0.489	1.141
ONL-ISL	1.244	1.021	1.647	0.896	0.791	1.025	-0.875	-1.008	-0.759
ILS-OLS	0.500	0.426	0.606	0.463	0.404	0.527	0.033	-0.091	0.162
OLS-RPE	0.721	0.541	1.011	0.604	0.490	0.743	0.447	0.249	0.624
RPE-Choroid	4.537	2.849	9.977	1.676	1.289	2.422	1.486	1.062	2.194
Average (μm)	1.513	1.052	2.612	0.884	0.717	1.138	0.007	-0.237	0.307

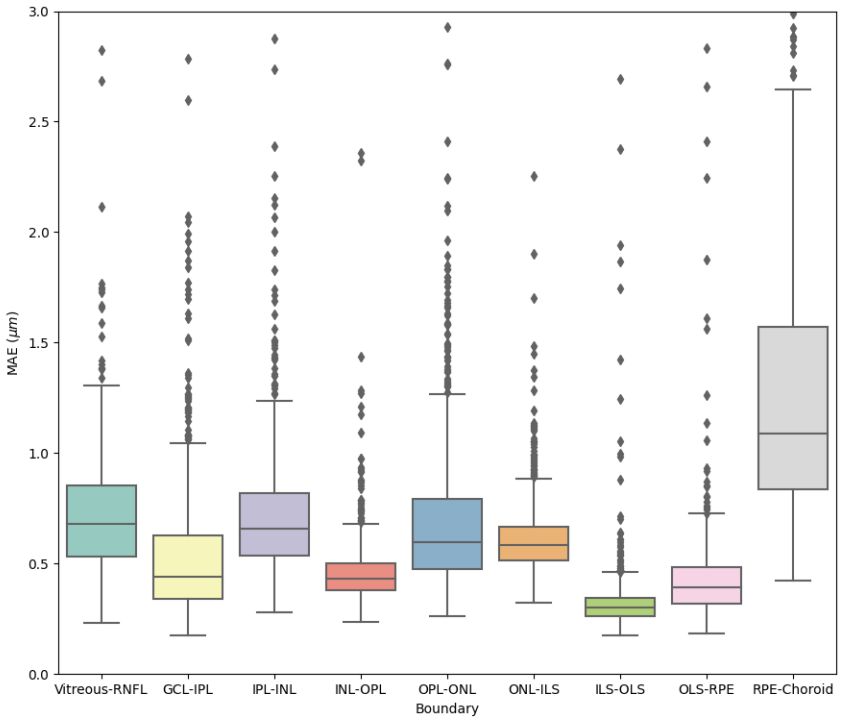


Figure 5.2: Box-plot of the mean absolute error distribution for each interface generated by the Attention-Res-U-Net model with the DS1 test set. The median, first, and third quartiles are shown. Whiskers at each quartile plus 1.5 times the interquartile range. Outliers are shown as diamonds.

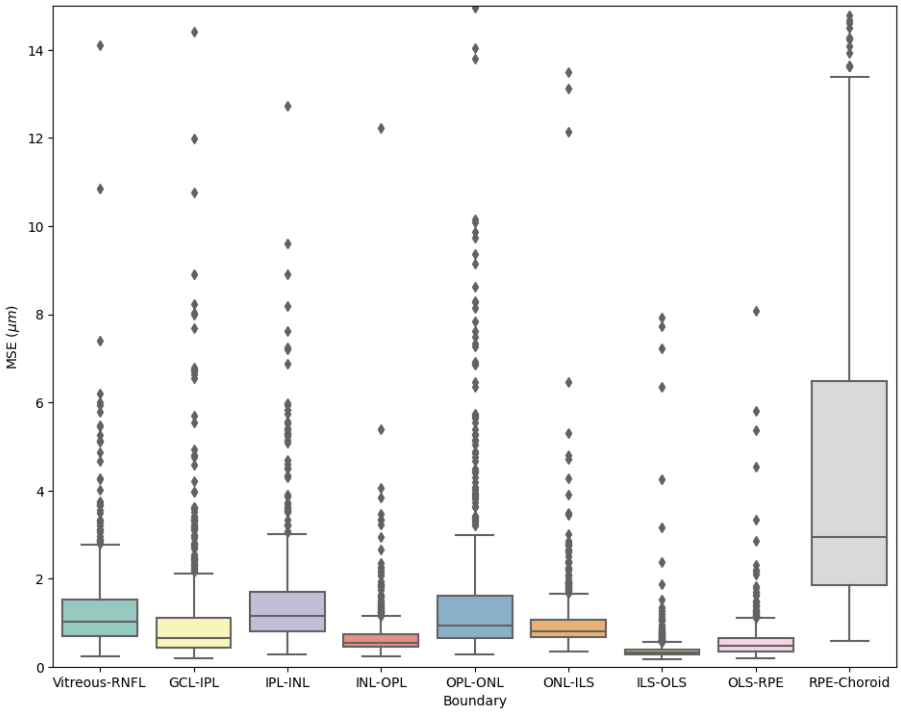


Figure 5.3: Box-plot of the mean squared error distribution for each interface generated by the Attention-Res-U-Net model with the DS1 test set. The median, first, and third quartiles are shown. Whiskers at each quartile plus 1.5 times the interquartile range. Outliers are shown as diamonds.

After analyzing the resulting segmentations and the obtained metrics, the algorithm's segmentation capability is noticeable. Figures 5.1 (a) and 5.1 (b) were picked to show the typical segmentation achieved by the model. Additional segmentations are showcased in Appendix A.

The proposed Attention-Res-U-Net achieved an average classification accuracy of 0.986 and an MAE of 0.884 μm. Hence, its capability to segment is evident not only visually, but also in quantitative terms, as the errors demonstrate satisfactory performance (as outlined in Table 5.2 and 5.1). The algorithm effectively achieves precise segmentation of the intended layers on the DS1 test set.

However, there are cases where the model presents weaknesses in its prediction ability, demonstrated in the selection of segmentations above (Figure 5.1 (c), (d)).

Firstly, it's clear that the RPE-Choroid boundary has the largest MAE, of 1.676 μm, as it's the boundary that presents the most faults when predicted, as depicted in Figure 5.1 (c) and Figure 5.1 (d), while the others present tighter MAE distributions (Figure 5.2).

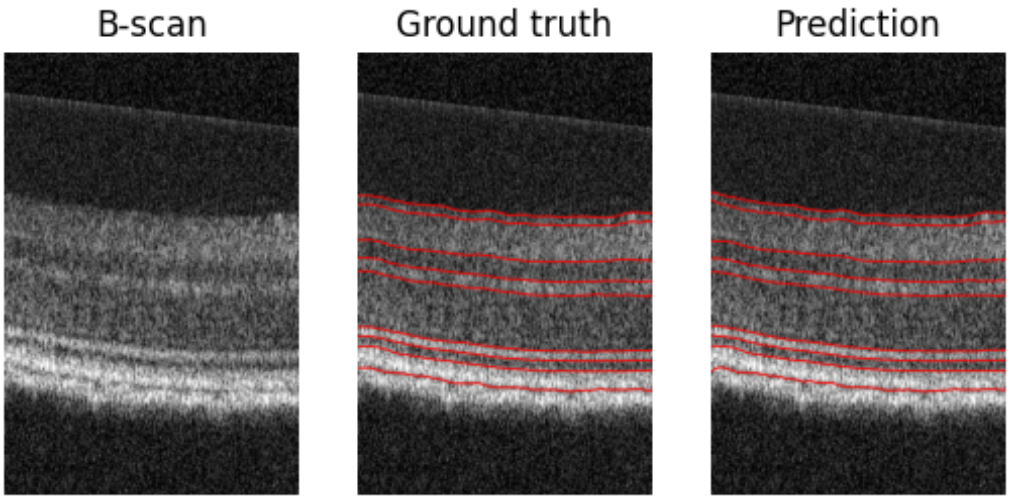
Furthermore, there are occurrences of manual segmentations on the dataset with unpredictable and random behavior, as shown in Figure 5.1(d). Given that most of the labels on the dataset follow a pattern, the model will make predictions based on what was learned through those regular labels. Therefore, the generated results will evidently differ from these sporadic manual labels. In addition, these labels may be inducing the algorithm in errors that can reflect in its performance.

Teacher-Student GAN

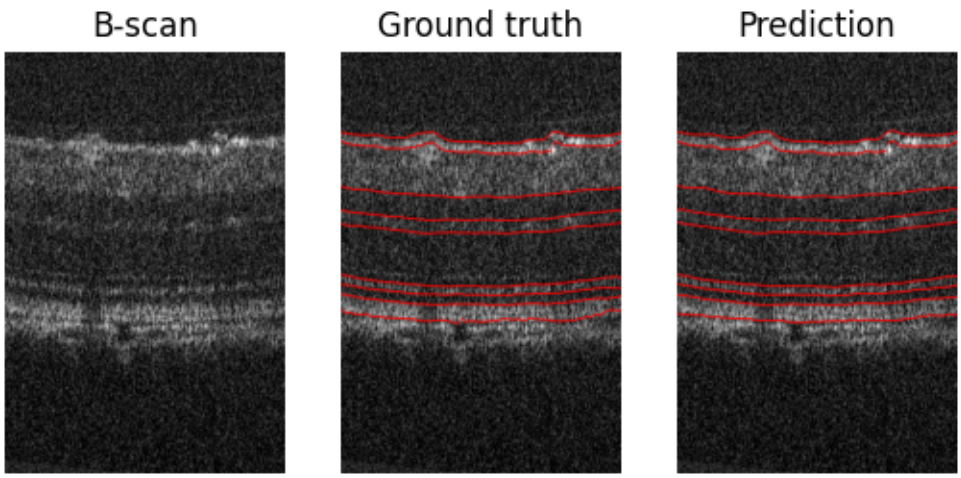
The training duration for this model was 36 hours and 7.73 minutes, spanning 100 epochs. Graphs illustrating the progression of loss can be found in Appendix A.

As the model underwent training, the loss consistently decreased. Eventually, it reached a stable value and approximately maintained this level throughout the remaining training period.

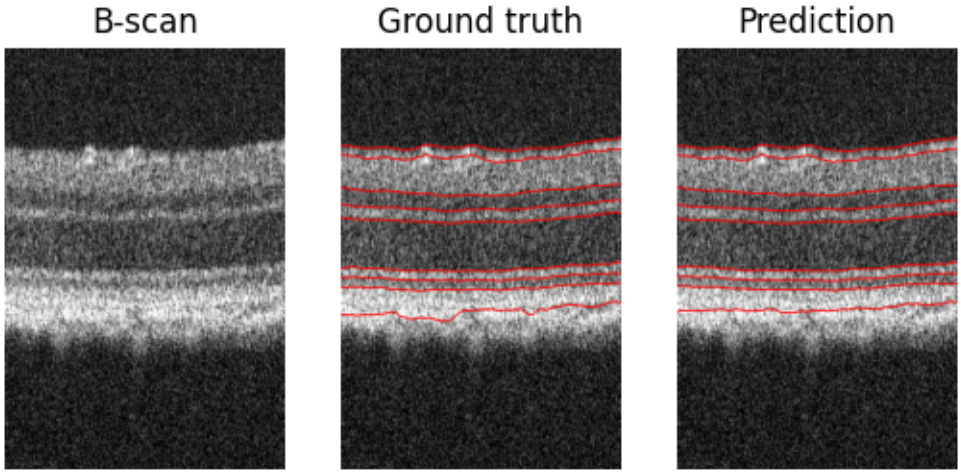
Even though a certain balance between the training tasks of the discriminator and generator was achieved, by considering different learning rates for each network, GANs are notorious for having unstable training dynamics. Achieving fast convergence can be challenging since they are always prone to diminished gradients, which slows down training, hence the longer training time.



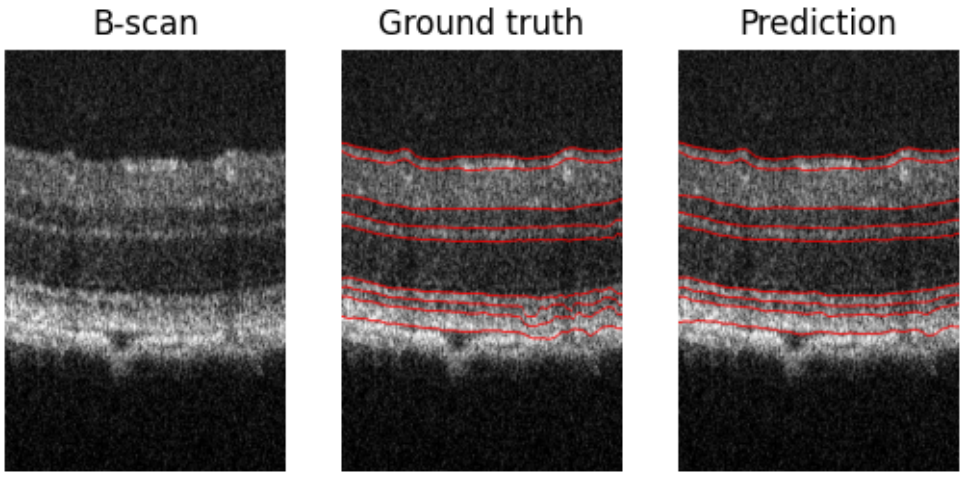
(a)



(b)



(c)



(d)

Figure 5.4: Selection of predicted layer interfaces by the Teacher-Student GAN, corresponding ground truth segmentations, and original B-scans for the DS1 test set.

Table 5.3: Performance metrics for the classification into layers of the DS1 test set using the Teacher-Student GAN model, along with the average and weighted average. The weighted average is pondered using the class weights, w_i.

	F1-Score			Dice Coefficient			Recall		
	Median	**Q1**	**Q3**	**Median**	**Q1**	**Q3**	**Median**	**Q1**	**Q3**
Vitreous	0.996	0.995	0.997	0.997	0.996	0.998	0.994	0.992	0.995
RNFL-GCL	0.973	0.961	0.980	0.973	0.961	0.980	0.987	0.978	0.992
IPL	0.992	0.989	0.993	0.990	0.988	0.992	0.990	0.985	0.992
INL	0.982	0.978	0.985	0.981	0.977	0.984	0.983	0.978	0.987
OPL	0.977	0.972	0.981	0.977	0.972	0.981	0.984	0.978	0.988
ONL	0.990	0.989	0.991	0.992	0.991	0.993	0.993	0.990	0.994
ILS	0.981	0.977	0.983	0.980	0.977	0.983	0.987	0.982	0.991
OLS	0.982	0.978	0.985	0.982	0.978	0.984	0.983	0.978	0.986
RPE	0.977	0.968	0.982	0.976	0.968	0.981	0.985	0.974	0.990
Choroid	0.998	0.996	0.998	0.998	0.996	0.998	0.997	0.995	0.998
Average	0.985	0.980	0.988	0.985	0.980	0.987	0.988	0.983	0.991
Weighted Average	0.985	0.981	0.988	0.985	0.981	0.988	0.989	0.983	0.991

	Precision			Accuracy			Intersect over Union		
	Median	**Q1**	**Q3**	**Median**	**Q1**	**Q3**	**Median**	**Q1**	**Q3**
Vitreous	1.000	0.999	1.000	0.994	0.992	0.995	0.994	0.992	0.996
RNFL-GCL	0.963	0.941	0.974	0.987	0.978	0.992	0.948	0.925	0.960
IPL	0.994	0.992	0.996	0.990	0.985	0.992	0.981	0.977	0.984
INL	0.982	0.976	0.986	0.983	0.978	0.987	0.962	0.954	0.968
OPL	0.972	0.965	0.976	0.984	0.978	0.988	0.955	0.945	0.962
ONL	0.988	0.987	0.989	0.993	0.990	0.994	0.984	0.982	0.986
ILS	0.976	0.970	0.979	0.987	0.982	0.991	0.962	0.954	0.966
OLS	0.982	0.977	0.985	0.983	0.978	0.986	0.964	0.957	0.969
RPE	0.974	0.964	0.981	0.985	0.974	0.99	0.953	0.938	0.963
Choroid	0.999	0.997	1.000	0.997	0.995	0.998	0.995	0.993	0.997
Average	0.983	0.977	0.987	0.988	0.983	0.991	0.970	0.962	0.975
Weighted Average	0.984	0.978	0.987	0.989	0.983	0.991	0.971	0.963	0.976

RNFL-GCL: Retinal Nerve Fiber Layer-Ganglion Cell Layer; IPL: Inner Plexiform Layer; INL: Inner Nuclear Layer; OPL: Outer Plexiform Layer; ONL: Outer Nuclear Layer; ILS: Photoreceptor Inner Segments; OLS: Photoreceptor Outer Segments; RPE: Retinal Pigment Epithelium.

Table 5.4: Distance errors per interface for the predicted segmentation by the Teacher-Student GAN model of the DS1 test set.

	Mean Squared Error (μm)			Mean Absolute Error (μm)			Mean Error (μm)		
	Median	**Q1**	**Q3**	**Median**	**Q1**	**Q3**	**Median**	**Q1**	**Q3**
Vitreous-RNFL	0.623	0.446	1.055	0.549	0.418	0.737	-0.375	-0.606	-0.157
GCL-IPL	0.559	0.392	1.097	0.484	0.362	0.689	0.135	-0.072	0.366
IPL-INL	0.949	0.694	1.359	0.688	0.575	0.854	-0.230	-0.467	0.032
INL-OPL	0.566	0.447	0.766	0.498	0.416	0.612	-0.168	-0.354	0.028
OPL-ONL	0.662	0.506	0.932	0.553	0.452	0.694	0.096	-0.099	0.391
ONL-ISL	0.414	0.348	0.538	0.398	0.338	0.488	-0.303	-0.410	-0.219
ILS-OLS	0.301	0.253	0.380	0.297	0.25	0.354	-0.078	-0.169	0.002
OLS-RPE	0.355	0.287	0.464	0.344	0.284	0.425	-0.053	-0.179	0.062
RPE-Choroid	2.287	1.293	5.474	1.045	0.784	1.617	0.281	-0.203	0.739
Average (μm)	0.746	0.518	1.341	0.540	0.431	0.719	-0.077	-0.284	0.138

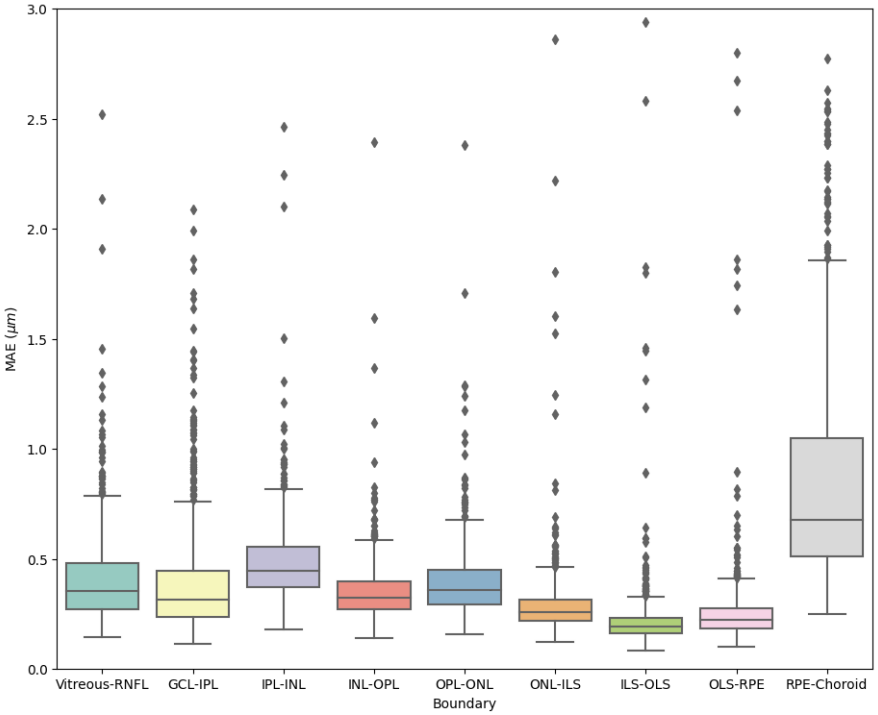


Figure 5.5: Box-plot of the mean absolute error distribution for each interface generated by the Teacher-Student GAN model with the DS1 test set. The median, first, and third quartiles are shown. Whiskers at each quartile plus 1.5 times the interquartile range. Outliers are shown as diamonds.

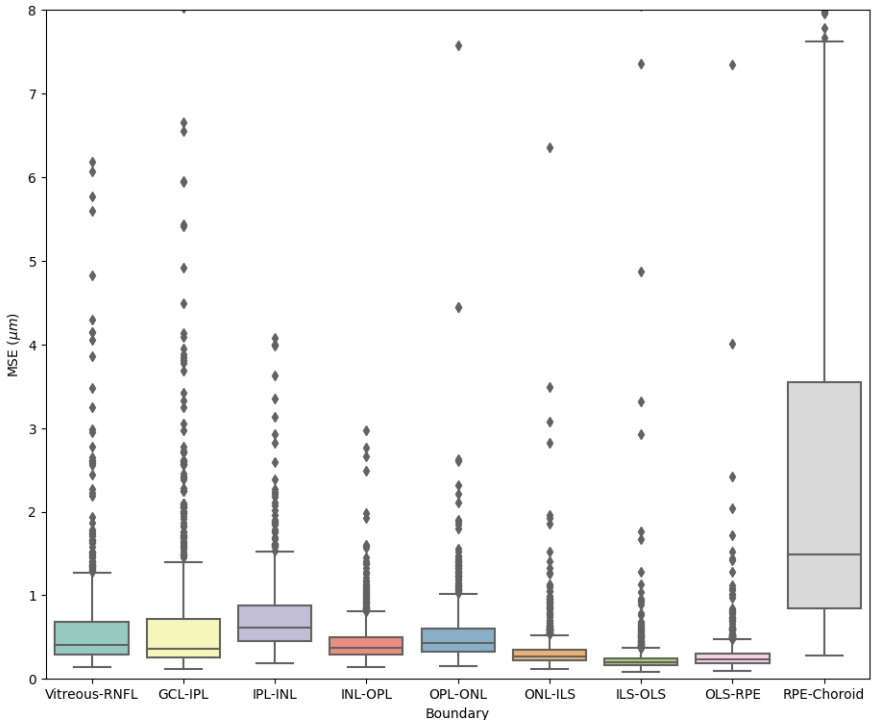


Figure 5.6: Box-plot of the mean squared error distribution for each interface generated by the Teacher-Student GAN model with the DS1 test set. The median, first, and third quartiles are shown. Whiskers at each quartile plus 1.5 times the interquartile range. Outliers are shown as diamonds.

The final segmentations and the computed metrics show that this model also accurately segments the provided B-Scans. Figures 5.4 (a) and 5.4 (b) were selected to illustrate the model's typical segmentation performance. More examples of segmentations are presented in Appendix A.

The Teacher-GAN network achieved an average classification accuracy of 0.989 and an MAE of 0.540 μm. Comparing Table 5.3 and 5.1, all classification metrics were improved. In addition, all the MAEs obtained were lower when compared to the previous model (as indicated in Table 5.4). This suggests the performance was improved (0.986 vs. 0.989 and 0.884 vs. 0.540 μm).

Similarly, the same inherent issues present in the Attention-Res-U-Net model are also observed in this current model, which is evidenced by the segmentations chosen above (Figure 5.4 (c), (d)).

However, the precision of the RPE-Choroid boundary was improved (Figure 5.4 (c), decreasing by 37.6% its MAE to 1.045 μm.

Generally, the results were more precise, which was reflected in all performance indicators, with particular reference to the IoU, which increased to 0.971, and the mean absolute errors of the boundaries which improved up to 71%.

Despite this, the challenges posed by the irregular hand-labeled ground truth images still persist, leading to the same segmentation inaccuracies, as depicted in Figure 5.4 (d).

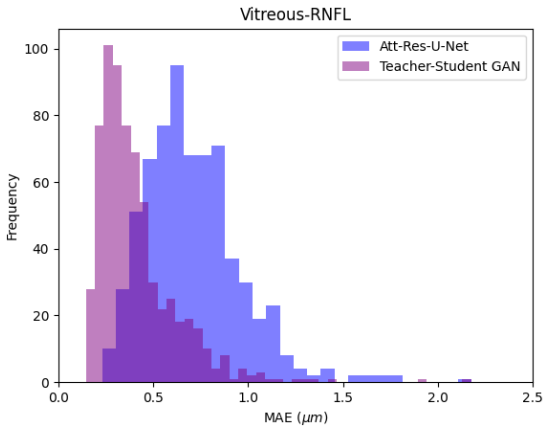
Attention-Res-U-Net vs. Teacher-Student GAN

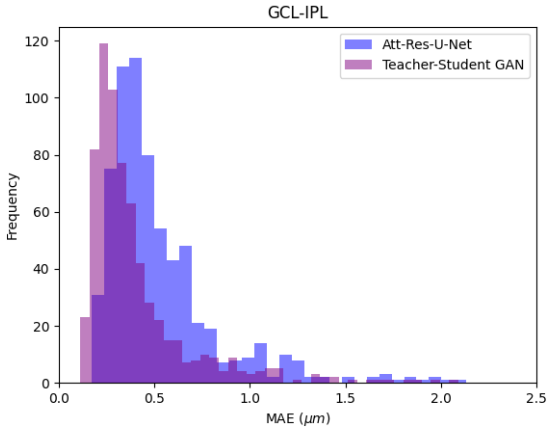
To compare the models, the Kolmogorov Smirnov (KS) test was first used to evaluate the normality of MAE distributions for each layer-to-layer interface.

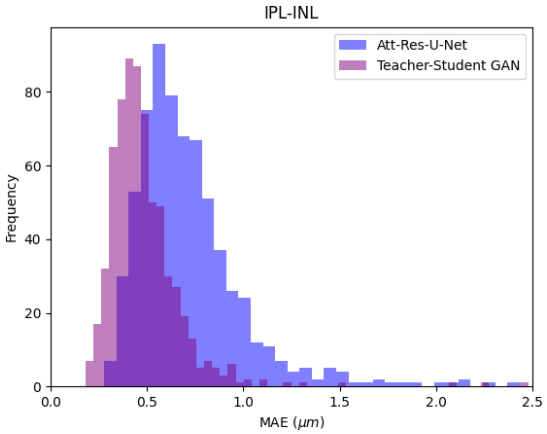
This statistical method is used to assess the similarity between the sample distribution and the reference distribution, in this case, a normal distribution. It produced a test statistic, often denoted as "D" or "KS statistic", along with the p-value. Table 5.5 shows the obtained results.

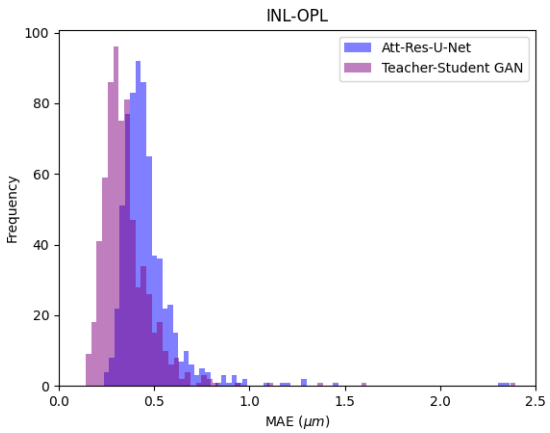
Table 5.5: Kolmogorov–Smirnov statistic and p-values for the KS test performed at every boundary regarding their mean absolute error distribution for each model.

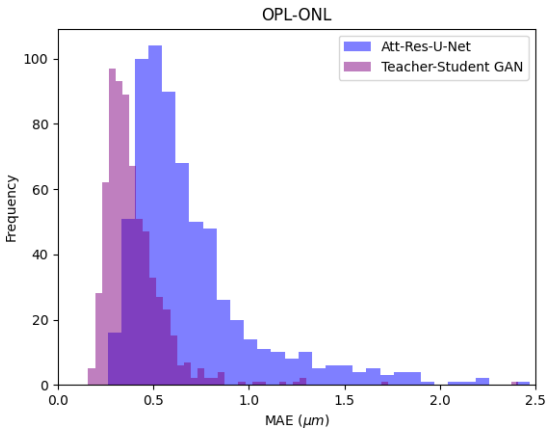
	Model			
	Attention-Res-U-Net		**Teacher-Student GAN**	
Boundary	***p-value***	**K-S statistic**	***p-value***	**K-S statistic**
Vitreous-RNFL	2.1251×10^{-238}	0.6063	1.9956×10^{-198}	0.5580
GCL-IPL	1.0764×10^{-208}	0.5710	8.2786×10^{-194}	0.5520
IPL-INL	9.7276×10^{-256}	0.6255	9.3669×10^{-217}	0.5809
INL-OPL	1.1208×10^{-231}	0.5986	7.1951×10^{-199}	0.5586
OPL-ONL	3.1504×10^{-246}	0.6151	1.0447×10^{-208}	0.5710
ONL-ILS	1.2966×10^{-270}	0.6413	1.7307×10^{-193}	0.5516
ILS-OLS	4.4511×10^{-209}	0.5715	2.6482×10^{-180}	0.5339
OLS-RPE	1.0485×10^{-210}	0.5735	1.0959×10^{-184}	0.5399
RPE-Choroid	1.2000×10^{-322}	0.6916	3.9112×10^{-239}	0.6071

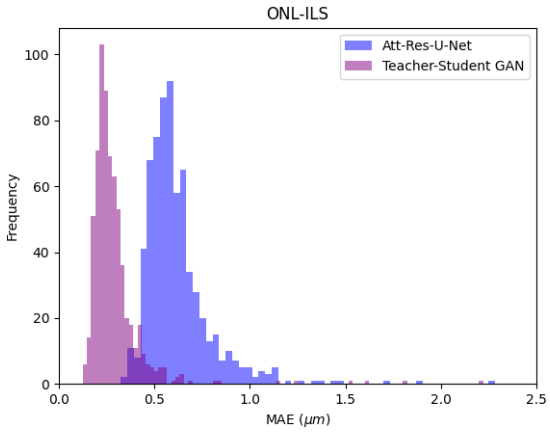

Vitreous-RNFL
Att-Res-U-Net
Teacher-Student GAN
Frequency
MAE (μm)

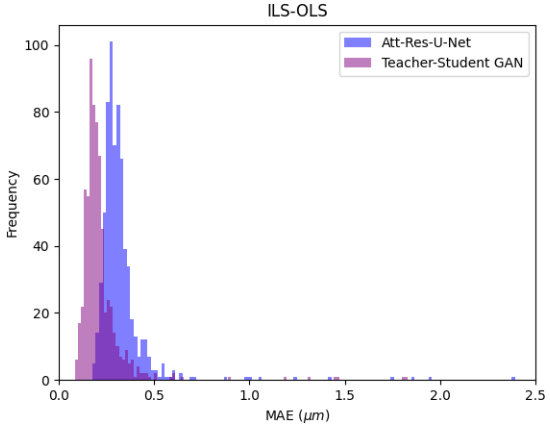

GCL-IPL
Att-Res-U-Net
Teacher-Student GAN
Frequency
MAE (μm)

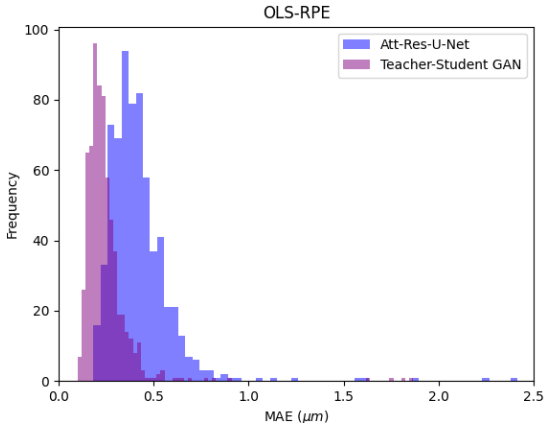

IPL-INL
Att-Res-U-Net
Teacher-Student GAN
Frequency
MAE (μm)


INL-OPL
Att-Res-U-Net
Teacher-Student GAN
Frequency
MAE (μm)


OPL-ONL
Att-Res-U-Net
Teacher-Student GAN
Frequency
MAE (μm)


ONL-ILS
Att-Res-U-Net
Teacher-Student GAN
Frequency
MAE (μm)


ILS-OLS
Att-Res-U-Net
Teacher-Student GAN
Frequency
MAE (μm)


OLS-RPE
Att-Res-U-Net
Teacher-Student GAN
Frequency
MAE (μm)

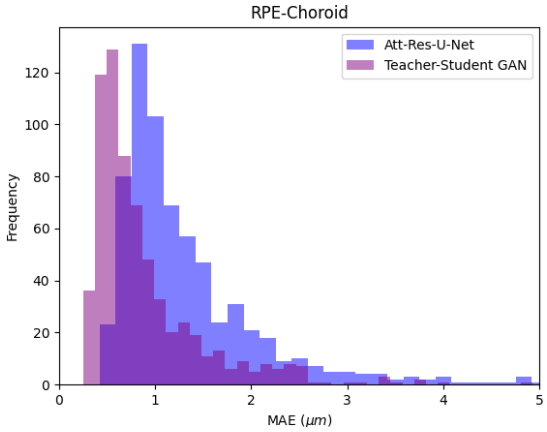


Figure 5.7: Distribution of the interfaces' mean absolute errors for each proposed model.

The chosen significance level, α, was 0.05. As registered above, all the *p-values* are below this value.

In parallel, the distribution graphs from Figure 5.7 visually accentuate this conclusion, as they all present a positive skew (right-skewed distribution).

Given the fact that all of the normality tests yielded statistically significant differences, the Wilcoxon test was selected to statistically compare both models. This non-parametric statistical method is used to compare two paired samples and determine if there is a significant difference between their distributions. In this case, the test was made for each interface under the following hypothesis:

$$\begin{cases} H_0: & med_{TS-GAN} \geq med_{Att-Res-U-Net} \\ H_a: & med_{TS-GAN} < med_{Att-Res-U-Net}, \end{cases} \tag{5.1}$$

where med is the median, $MAE_{TS-GAN} = med_{TS-GAN}$ and $MAE_{Att-Res-U-Net} = med_{Att-Res-U-Net}$.

Table 5.6 shows the obtained results.

Table 5.6: W statistic and p-values for the Wilcoxon test performed between the models at every boundary.

Boundary	***p-value***	**W Statistic**
Vitreous-RNFL	1.6035×10^{-105}	3665.0
GCL-IPL	2.9992×10^{-74}	21479.5
IPL-INL	5.4203×10^{-86}	14415.0
INL-OPL	3.3013×10^{-81}	17405.5
OPL-ONL	1.9107×10^{-103}	4591.0
ONL-ILS	4.3381×10^{-106}	3362.0
ILS-OLS	3.9189×10^{-101}	5938.5
OLS-RPE	1.6244×10^{-100}	6275.5
RPE-Choroid	3.0976×10^{-77}	19729.5

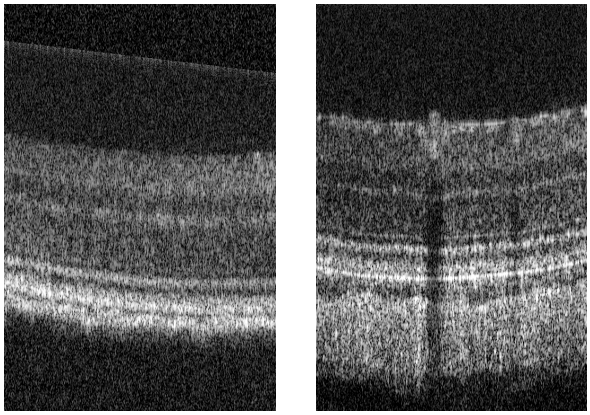
Based on the outcomes, with a selected α of 0.05, it is evident all the *p-values* fall below this threshold. Therefore, the null hypothesis H_0 can be rejected, and this can be interpreted as evidence that there is a significant difference between the compared distributions, in this case, that $MAE_{TS-GAN} < MAE_{Att-Res-U-Net}$.

In spite of the significant improvement achieved by the GAN model, training the generator and discriminator gave rise to convergence challenges, and attaining optimal solutions was not straightforward. Moreover, the memory usage of this model is notably higher, and the training and testing duration is much larger when compared to that of the standalone Attention-Res-U-Net. In fact, the testing time per B-scan is 1.47 s for the Attention-Res-U-Net and 2.13 s for the Teacher-Student GAN. Due to the time constraints associated with a master's thesis, time was an important factor to consider. Furthermore, the clinical significance of the improvement is arguable. The difference in mean absolute error was only about 0.34 μm, a value below the axial resolution of the system. Thus, the Attention-Res-U-Net model was selected as the main model and only the Attention-Res-U-Net model was used for datasets 2 and 3.

5.1.2 Dataset 2 - DS2

This sub-section focuses on showcasing and analyzing the segmentation results achieved using the Attention-Res-U-Net on the DS2 dataset, composed of WT and type 1 diabetes rats B-scans. Once again, additional segmentations are showcased in Appendix A.

In DS2, the retinas of type 1 diabetes rats were imaged. Since the retinas are from a different animal (mice vs. rat) and model of disease, their structure will be visually different in the collected B-scans. Because of this, there are interfaces in the DS2 dataset that cannot be identified in the DS1 dataset used for training the network, as can be seen in Figure 5.9. In this evaluation, only the common interfaces between the two datasets were considered. Interfaces ONL-ILS and OLS-RPE were removed.



(a) (b)

Figure 5.8: A B-scan example from the DS1 dataset (a) and a B-scan example from the DS2 dataset (b).

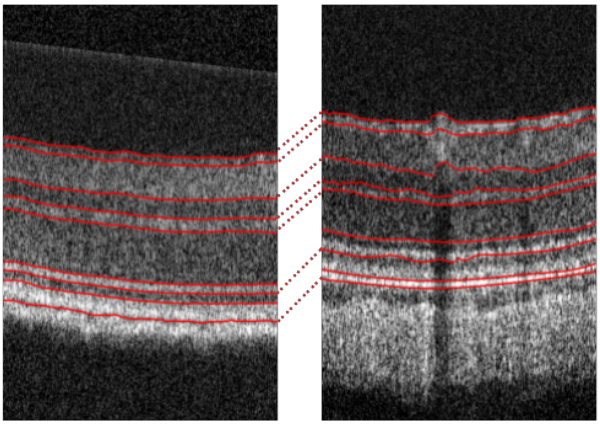
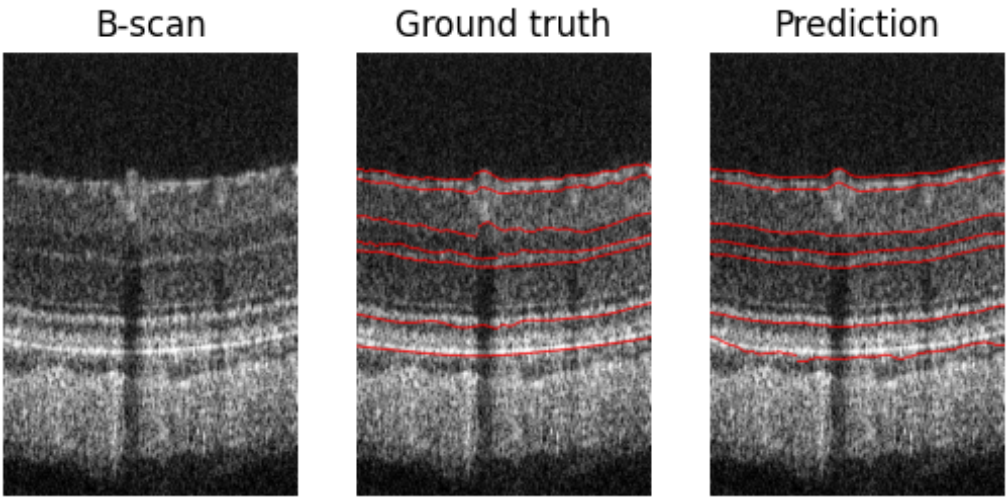
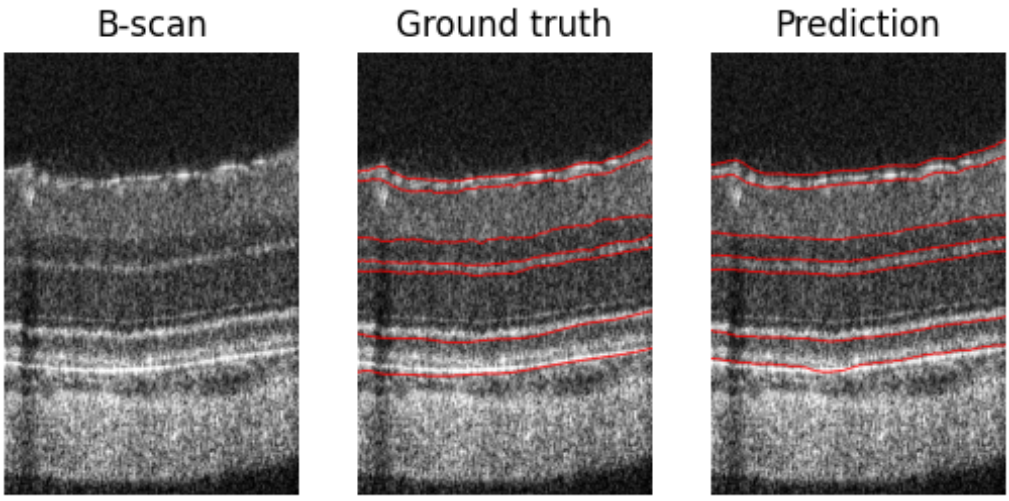


Figure 5.9: A segmentation from the DS1 dataset (a) and a segmentation from the DS2 dataset (b). The common interfaces are connected in red between both B-scans.

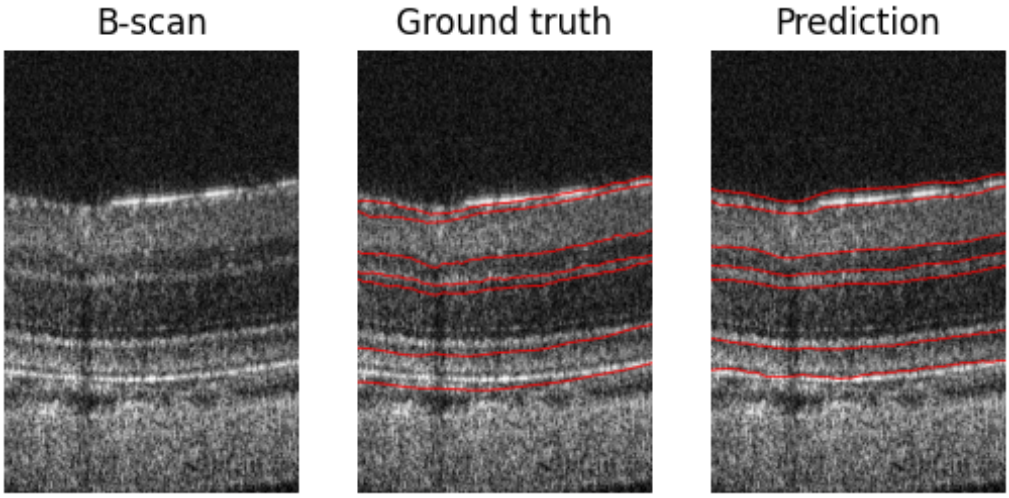
Presented below is a selection of segmentation outputs alongside their corresponding ground truth. Additionally, tables are supplied to summarize classification errors and boundary distance errors, supplemented by visual plots, allowing a detailed examination of the results.



(a)



(b)



(c)

Figure 5.10: Selection of predicted layer interfaces by the Attention-Res-U-Net, corresponding ground truth segmentations, and original B-scans for the DS2 test set.

Table 5.7: Performance metrics for the classification into layers of the DS2 test set using the Attention-Res-U-Net model, along with the average and weighted average. The weighted average is pondered using the class weights, w_i.

	F1-Score			**Dice Coefficient**			**Recall**		
	Median	**Q1**	**Q3**	**Median**	**Q1**	**Q3**	**Median**	**Q1**	**Q3**
Vitreous	0.987	0.978	0.99	0.987	0.979	0.990	0.975	0.968	0.984
RNFL-GCL	0.753	0.616	0.806	0.722	0.611	0.797	0.833	0.728	0.867
IPL	0.949	0.924	0.958	0.938	0.917	0.949	0.963	0.949	0.974
INL	0.882	0.825	0.907	0.855	0.812	0.879	0.813	0.781	0.871
OPL	0.788	0.754	0.823	0.711	0.651	0.755	0.950	0.900	0.967
Choroid	0.965	0.936	0.982	0.949	0.924	0.960	0.974	0.928	0.994
Average	0.887	0.839	0.911	0.860	0.816	0.888	0.918	0.876	0.943
Weighted Average	0.887	0.840	0.911	0.860	0.817	0.889	0.919	0.876	0.943

	Precision			**Accuracy**			**Intersect over Union**		
	Median	**Q1**	**Q3**	**Median**	**Q1**	**Q3**	**Median**	**Q1**	**Q3**
Vitreous	1.000	1.000	1.000	0.975	0.968	0.984	0.975	0.958	0.980
RNFL-GCL	0.675	0.501	0.784	0.833	0.728	0.867	0.565	0.440	0.662
IPL	0.929	0.885	0.954	0.963	0.949	0.974	0.884	0.846	0.902
INL	0.953	0.910	0.961	0.813	0.781	0.871	0.747	0.684	0.785
OPL	0.686	0.628	0.722	0.950	0.900	0.967	0.552	0.483	0.606
Choroid	0.978	0.962	0.998	0.974	0.928	0.994	0.903	0.859	0.924
Average	0.870	0.814	0.903	0.918	0.876	0.943	0.771	0.712	0.810
Weighted Average	0.871	0.815	0.903	0.918	0.876	0.943	0.772	0.713	0.811

RNFL-GCL: Retinal Nerve Fiber Layer-Ganglion Cell Layer; IPL: Inner Plexiform Layer; INL: Inner Nuclear Layer; OPL: Outer Plexiform Layer; RPE: Retinal Pigment Epithelium.

Table 5.8: Distance errors per interface for the predicted segmentation by the Attention-Res-U-Net model of the DS2 test set.

	Mean Squared Error (μm)			**Mean Absolute Error** (μm)			**Mean Error** (μm)		
	Median	**Q1**	**Q3**	**Median**	**Q1**	**Q3**	**Median**	**Q1**	**Q3**
Vitreous-RNFL	54.356	34.789	126.236	7.023	5.092	10.818	-6.074	-8.947	-4.551
GCL-IPL	24.186	20.436	46.648	4.209	3.699	6.455	-3.441	-4.773	-0.404
IPL-INL	21.160	8.986	40.115	3.656	2.406	5.828	0.342	-1.625	2.293
INL-OPL	24.754	13.139	48.764	4.008	3.139	6.279	-3.145	-6.047	-1.660
OPL-ONL	38.346	11.889	46.697	4.555	2.844	6.084	3.326	0.844	5.813
ILS-OLS	25.192	9.805	43.723	4.111	2.732	6.231	1.664	-0.109	3.967
RPE-Choroid	82.035	45.879	250.516	8.094	5.627	14.520	-2.936	-7.861	4.356
Average (μm)	38.576	20.703	86.100	5.094	3.648	8.031	-1.466	-4.074	1.402

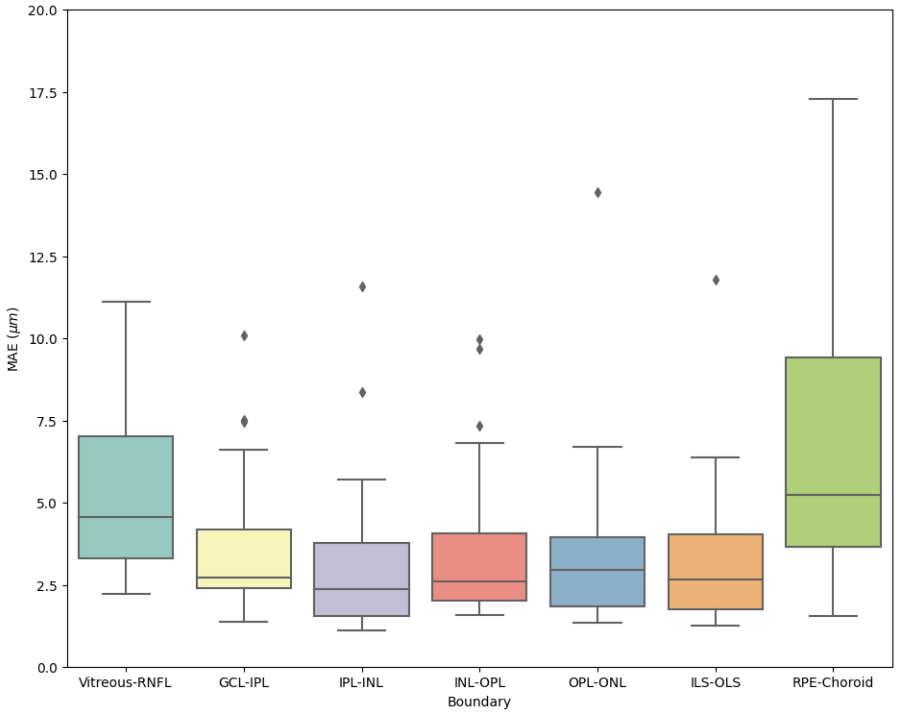


Figure 5.11: Box-plot of the mean absolute error distribution for each interface generated by the Attention-Res-U-Net model with the DS2 test set. The median, first, and third quartiles are shown. Whiskers at each quartile plus 1.5 times the interquartile range. Outliers are shown as diamonds.

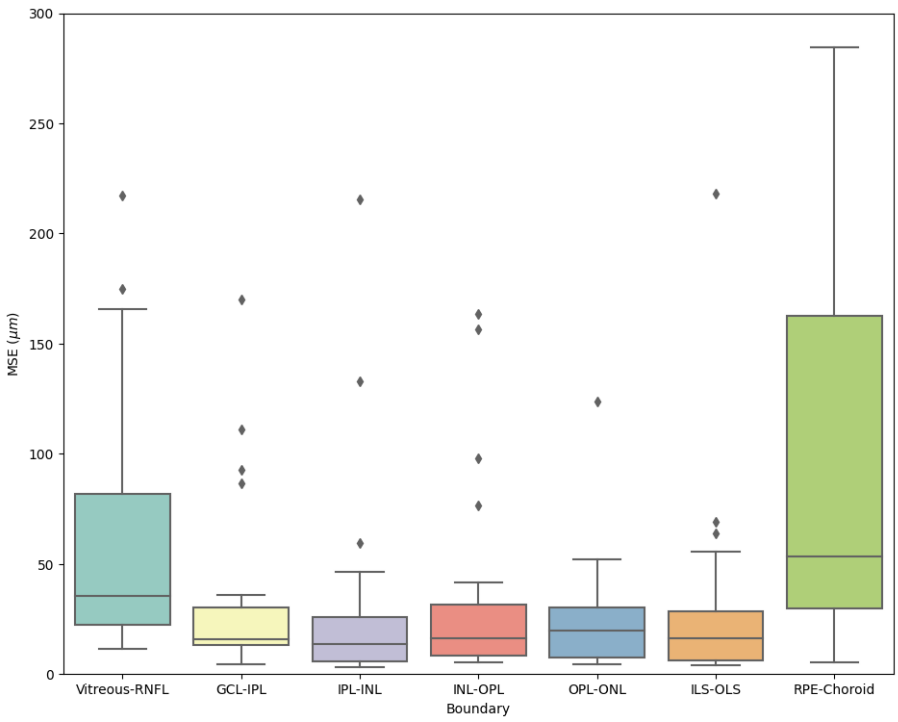


Figure 5.12: Box-plot of the mean squared error distribution for each interface generated by the Attention-Res-U-Net model with the DS2 test set. The median, first, and third quartiles are shown. Whiskers at each quartile plus 1.5 times the interquartile range. Outliers are shown as diamonds.

After analyzing the resulting segmentations and the obtained metrics, the algorithm does not perform as well in this dataset. Figures 5.10 (a), 5.10 (b), and 5.10 (c) show the typical segmentation achieved by the model.

The proposed Attention-Res-U-Net achieved an average classification accuracy of 0.918 and an MAE of 5.094 μm. A direct comparison with the averages from DS1 is not possible because the same interfaces were not considered. Nevertheless, the difference suggests that the model performs worst on DS2 (an accuracy of 0.986 vs. 0.918 and MAE of 0.884 vs. 5.094 μm). A direct comparison per layer is possible. For all individual layers and interfaces considered, classification accuracy is lower (Table 5.7), and the mean absolute errors are higher than in DS1, as demonstrated in Table 5.8, indicating the model performs worst on DS2 compared to DS1. These results are expected since the latter was completely

independent. The animal model was different, as was the imaging protocol and the expert responsible for the manual segmentation. However, these results are extremely important because they are representative of real-world performance. In fact, even two different human experts given a similar segmentation task will disagree on the exact location of the interfaces [109].

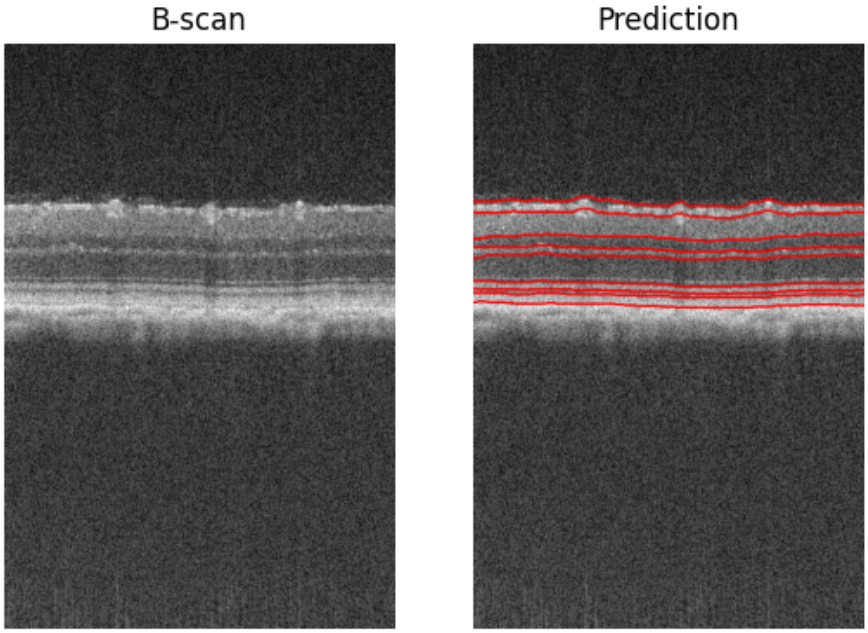
Visually (Figure 5.10), the proposed approach appears to correctly predict the interfaces in the majority of the cases. However, it is possible that it could benefit from transfer learning when considering a new animal model. This is not without drawbacks. Transfer learning implies retraining with all its caveats. Most importantly, manual segmentation is needed to create a new ground truth.

It is worth noting that the RPE-Choroid interface exhibits the largest error in both DS2 and DS1 for the Attention-Res-U-Net and Teacher-Student GAN networks. This could be attributed to the high reflectivity of the RPE layer in the B-scans, which may cause the transition between the RPE and the choroid to be less clear and more ambiguous in certain cases (Figure 5.1 (c)).

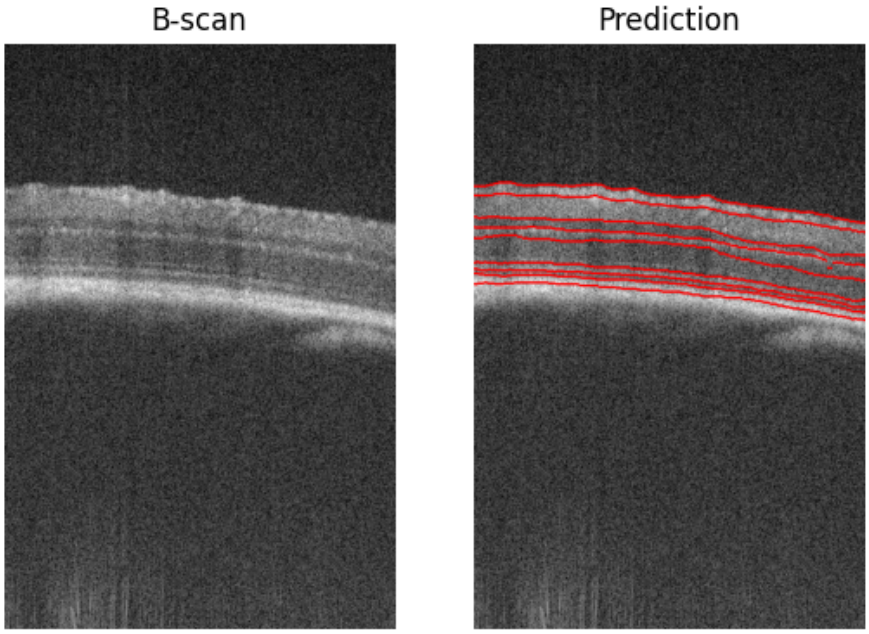
5.2 Bioptigen Dataset 3 - DS3

In this subsection, the segmentation results of the dataset obtained from the Bioptigen system, DS3, will be presented and analyzed.

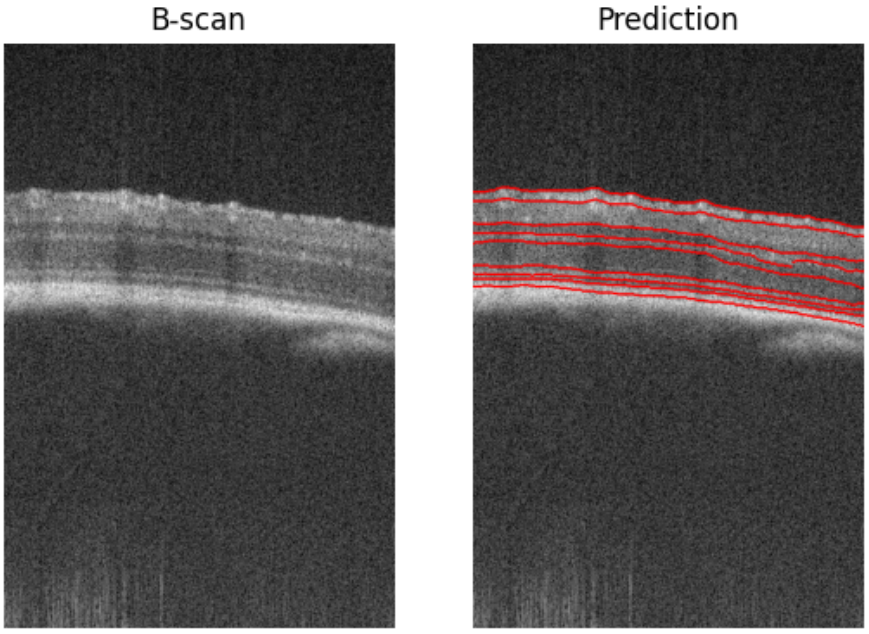
Attention-Res-Unet was retrained by transfer leaning for 50 epochs (8 minutes and 16 seconds). Displayed below is a selection of B-scans overlayed with the predicted interfaces. No ground truth existed for the test set. Thus, no quantitative results are shown.



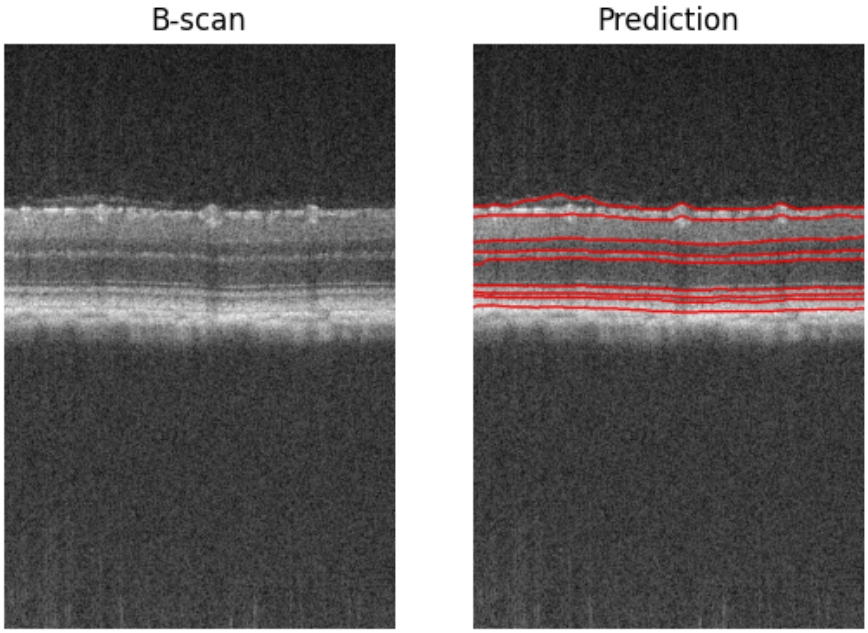
(a)



(b)



(c)



(d)

Figure 5.13: Selection of predicted layer interfaces by the retrained Attention-Res-U-Net model (right) and corresponding B-scan (left) for the DS3 test set.

After a visual examination of the results, supplemented by expert assessment, it is noteworthy that while there are no specific metrics available for comparison with the ground truth labels, the algorithm appears to correctly segment B-scans from the Bioptigen system. This can be seen in Figure 5.13 (a) and Figure 5.13 (b).

Nonetheless, there are segmentation flaws related to defects located on the limits of the OPL layer, as shown in Figure 5.13 (d).

Besides this, in the Vitreous-RNFL boundary, there are segmentation errors due to the presence of noise above the RNFL-GCL layer in some of the scans, as depicted in Figure 5.13 (c).

Most likely, these issues could be mitigated if a more extensive, diverse, and higher-quality training dataset were to be used in the transfer learning process, that allowed the model to learn above the noise.

Chapter 6

Conclusion and Future Work

The main focus of this study was the segmentation of retinal layers in B-scans collected from various animal models of disease and imaging acquisition systems. In the literature, research in this field has predominantly focused on addressing segmentation challenges within specific datasets, each using a single acquisition system. Therefore, it is essential to develop segmentation methods that can perform in a set of eyes from distinct wild-type and animal models of disease and acquisition systems. This dissertation attempted to provide a deep-learning solution for this issue.

The objectives defined at the beginning of this paper were achieved. To accomplish them, two convolutional neural networks were developed to segment retinal layers: the Attention-Res-U-Net and the Teacher-Student GAN. Each architecture was studied to create flexible models capable of correctly segmenting the intended layers in a diversity of B-scans, each presenting different characteristics.

The introduction of the attention mechanism and the weighted focal loss function were crucial, as they mitigated the class imbalance issue, improving the segmentation performance on thinner layers and, mostly, helped the models to focus equally on every layer and the most relevant regions of the layers, particularly the interfaces, towards a more accurate segmentation.

As for the results, the system performed well on DS1, collected on the Phoenix OCT system. The Attention-Res-U-Net model got an MAE between the predicted and actual

interfaces of 0.88 μm with an interquartile range (IQR) of 0.42 μm and the Teacher-Student GAN an MAE of 0.54 μm with an IQR of 0.29 μm. Here, the Teacher Student GAN outperformed the Attention-Res-U-Net. The PatchGAN (discriminator) essentially refined the generator's performance, which was Attention-Res-U-Net based, by providing constant feedback on the generated predictions on a patch level.

For the DS2 dataset, also collected from the Phoenix OCT, the Attention-Res-U-Net obtained an average MAE of 5.09 μm with an IQR of 4.38 μm, which was considerably higher. Even with a more significant error, the algorithm was able to segment the B-scan to some extent.

Finally, even though no metrics were associated, an expert assessed the achieved results to conclude that the Attention-Res-U-Net also showed satisfactory segmentation results on the Bioptigen dataset (DS3).

Overall, the algorithm demonstrated adaptable segmentation capabilities when presented with new data from different models of disease, specifically Alzheimer's disease and diabetes, from other acquisition systems, including the Phoenix OCT and the Bioptigen. This work contributed to flexible retinal segmentation methods that can be applied to various models of disease (mice and rats) and OCT acquisition systems.

However, there are some hurdles and limitations to the proposed solution. First, the memory constraints made experimenting and, above all, training the models challenging. Using the entire training dataset proved difficult, even with the GPUs at hand. This issue was overcome by loading the dataset in subsets instead of the whole dataset simultaneously, reducing the memory requirement. At the same time, training CNNs is a demanding process that requires experimentation and a significant time investment.

Furthermore, it was shown that CNNs segmentations may differ from the ground truth segmentations. This does not indicate a poor performance. Indeed, manual segmentations are subjective and can vary among annotators.

As for future improvements, a more robust hyperparameter tuning process may be added to optimize the hyperparameters, such as a grid search. Furthermore, a transfer learning approach using a more extensive and diverse training dataset could have boosted

the model's performance for the Bioptigen system but also for different animal models of disease. Providing the model with more diverse training data would perhaps enhance its segmentation flexibility and generalization capability. Also, it will be interesting to investigate further the potential of the generative adversarial network.

Bibliography

[1] Andrzej Grzybowski, Piero Barboni, et al. *OCT and imaging in central nervous system diseases: the eye as a window to the brain*, volume 2020. Springer, 2020.

[2] Joshua Ong, Arman Zarnegar, Giulia Corradetti, Sumit Randhir Singh, and Jay Chhablani. Advances in optical coherence tomography imaging technology and techniques for choroidal and retinal disorders, 2022.

[3] https://www.who.int/news-room/fact-sheets/detail/dementia, Mar 2023.

[4] Pooja Godara, Adam M Dubis, Austin Roorda, Jacque L Duncan, and Joseph Carroll. Adaptive optics retinal imaging: emerging clinical applications. *Optometry and vision science: official publication of the American Academy of Optometry*, 87(12):930, 2010.

[5] Niall Patton, Tariq M Aslam, Thomas MacGillivray, Ian J Deary, Baljean Dhillon, Robert H Eikelboom, Kanagasingam Yogesan, and Ian J Constable. Retinal image analysis: concepts, applications and potential. *Progress in retinal and eye research*, 25(1):99–127, 2006.

[6] David Huang, Eric A Swanson, Charles P Lin, Joel S Schuman, William G Stinson, Warren Chang, Michael R Hee, Thomas Flotte, Kenton Gregory, Carmen A Puliafito, and James G Fujimoto. Optical coherence tomography. *Science*, 254:1178–1181, 1991.

[7] https://www.leica-microsystems.com/applications/medical/optical-coherence-tomography-oct/, Sep 2023.

[8] https://phoenixmicron.com/phoenix-micron-retinal-camera/micron-iv/, Apr 2023.

[9] Michael D Abràmoff, Mona K Garvin, and Milan Sonka. Retinal imaging and image analysis. *IEEE reviews in biomedical engineering*, 3:169–208, 2010.

[10] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A W M van der Laak, Bram van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, 2017.

[11] Nithya Rajagopalan, Venkateswaran N., Alex Noel Josephraj, and Srithaladevi E. Diagnosis of retinal disorders from optical coherence tomography images using cnn, 2021.

[12] Dale Purves and Stephen Mark Williams. *Neuroscience. 2nd edition.* Sinauer Associates, 2001.

[13] A Dick. Basic and clinical science course: complete set, 2007.

[14] Emily Dawn Cole, Eduardo Amorim Novais, Ricardo Noguera Louzada, and Nadia K Waheed. Contemporary retinal imaging techniques in diabetic retinopathy: a review. *Clinical & experimental ophthalmology*, 44(4):289–299, 2016.

[15] P Hariharan and P Hariharan. *3 - Two-Beam Interferometers*, pages 13–22. Academic Press, second edition edition, 2007.

[16] A F Fercher, C K Hitzenberger, G Kamp, and S Y El-Zaiat. Measurement of intraocular distances by backscattering spectral interferometry. *Optics Communications*, 117:43–48, 1995.

[17] P Hariharan and P Hariharan. *16 - Fourier Transform Spectroscopy*, pages 145–151. Academic Press, second edition edition, 2007.

[18] Michelle L Gabriele, Gadi Wollstein, Hiroshi Ishikawa, Larry Kagemann, Juan Xu, Lindsey S Folio, and Joel S Schuman. Optical coherence tomography: History,

current status, and laboratory work. *Investigative Ophthalmology Visual Science*, 52:2425–2436, 4 2011.

[19] Maciej Wojtkowski, Vivek Srinivasan, James G Fujimoto, Tony Ko, Joel S Schuman, Andrzej Kowalczyk, and Jay S Duker. Three-dimensional retinal imaging with high-speed ultrahigh-resolution optical coherence tomography. *Ophthalmology*, 112:1734 – 1746, 2005. Cited by: 583; All Open Access, Green Open Access.

[20] Vivek J Srinivasan, Maciej Wojtkowski, Andre J Witkin, Jay S Duker, Tony H Ko, Mariana Carvalho, Joel S Schuman, Andrzej Kowalczyk, and James G Fujimoto. High-definition and 3-dimensional imaging of macular pathologies with high-speed ultrahigh-resolution optical coherence tomography. *Ophthalmology*, 113:2054–2065.e3, 2006.

[21] Suhail Alam, Robert J Zawadzki, Stacey Choi, Christina Gerth, Susanna S Park, Lawrence Morse, and John S Werner. Clinical application of rapid serial fourier-domain optical coherence tomography for macular imaging. *Ophthalmology*, 113:1425 – 1431, 2006. Cited by: 153; All Open Access, Green Open Access.

[22] Ana Batista, Pedro Guimarães, João Martins, Paula I Moreira, António Francisco Ambrósio, Miguel Castelo-Branco, Pedro Serranho, and Rui Bernardes. Normative mice retinal thickness: 16-month longitudinal characterization of wild-type mice and changes in a model of alzheimer's disease. *Frontiers in Aging Neuroscience*, 15, 2023.

[23] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.

[24] Keiron and Nash Ryan O'Shea. An introduction to convolutional neural networks. *International Journal for Research in Applied Science and Engineering Technology*, 10, 2015.

[25] R. Sathya and Annamma Abraham. Comparison of supervised and unsupervised learning algorithms for pattern classification. *International Journal of Advanced Research in Artificial Intelligence*, 2, 2013.

[26] B. W. White and Frank Rosenblatt. Principles of neurodynamics: Perceptrons and the theory of brain mechanisms. *The American Journal of Psychology*, 76:705, 12 1963.

[27] Prateek. Statistics is freaking hard: Wtf is activation function. https://towardsdatascience.com/statistics-is-freaking-hard-wtf-is-activation-function-df8342cdf292, Aug 2017.

[28] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[29] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[30] Mehmet Akar, Mahmut Hekim, and Umut Orhan. Mechanical fault detection in permanent magnet synchronous motors using equal width discretization-based probability distribution and a neural network model. *TURKISH JOURNAL OF ELECTRICAL ENGINEERING COMPUTER SCIENCES*, 23:813–823, 01 2015.

[31] Raul Rojas and Raúl Rojas. The backpropagation algorithm. *Neural networks: a systematic introduction*, pages 149–182, 1996.

[32] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[33] Leslie N Smith. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*, 2018.

[34] Simon Haykin. *Redes neurais: princípios e prática*. Bookman Editora, 2001.

[35] Qi Wang, Yue Ma, Kun Zhao, and Yingjie Tian. A comprehensive survey of loss functions in machine learning. *Annals of Data Science*, pages 1–26, 2020.

[36] Yaoshiang Ho and Samuel Wookey. The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. *IEEE access*, 8:4806–4813, 2019.

[37] Jie Cao, Zhe Su, Liyun Yu, Dongliang Chang, Xiaoxu Li, and Zhanyu Ma. Softmax cross entropy loss with unbiased decision boundary for image classification. In *2018 Chinese automation congress (CAC)*, pages 2028–2032. IEEE, 2018.

[38] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 2022.

[39] Andrinandrasana David Rasamoelina, Fouzia Adjailia, and Peter Sinčák. A review of activation function for artificial neural network. In *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 281–286. IEEE, 2020.

[40] Mohit Goyal, Rajan Goyal, P Venkatappa Reddy, and Brejesh Lall. Activation functions. *Deep learning: Algorithms and applications*, pages 1–30, 2020.

[41] Tomasz Szandała. Review and comparison of commonly used activation functions for deep neural networks. *Bio-inspired neurocomputing*, pages 203–224, 2021.

[42] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Atlanta, GA, 2013.

[43] Bolin Gao and Lacra Pavel. On the properties of the softmax function with application in game theory and reinforcement learning. *arXiv preprint arXiv:1704.00805*, 2017.

[44] Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196, 1993.

[45] Sarit Khirirat, Hamid Reza Feyzmahdavian, and Mikael Johansson. Mini-batch gradient descent: Faster convergence under data sparsity. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2880–2887. IEEE, 2017.

[46] Agnes Lydia and Sagayaraj Francis. Adagrad—an optimizer for stochastic gradient descent. *Int. J. Inf. Comput. Sci*, 6(5):566–568, 2019.

[47] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[48] Tijmen Tieleman, Geoffrey Hinton, et al. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[49] Saad Hikmat Haji and Adnan Mohsin Abdulazeez. Comparison of optimization techniques based on gradient descent algorithm: A review. *PalArch's Journal of Archaeology of Egypt/Egyptology*, 18(4):2715–2743, 2021.

[50] Xue Ying. An overview of overfitting and its solutions. *Journal of Physics: Conference Series*, 1168(2):022022, feb 2019.

[51] Strahinja Zivkovic. 018 pytorch - popular techniques to prevent the overfitting in a neural networks. https://datahacker.rs/018-pytorch-popular-techniques-to-prevent-the-overfitting-in-a-neural-networks/, Nov 2021.

[52] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[53] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):6999–7019, 2022.

[54] Aseem Patil and Milind Rane. Convolutional neural networks: An overview and its applications in pattern recognition. In Tomonobu Senjyu, Parikshit N. Mahalle,

Thinagaran Perumal, and Amit Joshi, editors, *Information and Communication Technology for Intelligent Systems*, pages 21–30, Singapore, 2021. Springer Singapore.

[55] Leyuan Fang, David Cunefare, Chong Wang, Robyn H Guymer, Shutao Li, and Sina Farsiu. Automatic segmentation of nine retinal layer boundaries in oct images of non-exudative amd patients using deep learning and graph search. *Biomed. Opt. Express*, 8:2732–2744, 5 2017.

[56] Kevin Gurney. *An introduction to neural networks*. CRC press, 2018.

[57] Janosh Riebesell. Convolution operator, Apr 2022.

[58] V Lakkavaram, Lakkavaram Raghuveer, Chatarband Kumar, G Sri, and Shaik Habeeb. A review on practical diagnostic of tomato plant diseases. 9 2019.

[59] `https://docs.nvidia.com/deeplearning/performance/dl-performance-fully-connected/index.html`, 2023.

[60] Michael R Hee, Joseph A Izatt, Eric A Swanson, David Huang, Joel S Schuman, Charles P Lin, Carmen A Puliafito, and James G Fujimoto. Optical coherence tomography of the human retina. *Archives of Ophthalmology*, 113:325–332, 3 1995.

[61] D Koozekanani, K Boyer, and C Roberts. Retinal thickness measurements from optical coherence tomography using a markov boundary model. *IEEE Transactions on Medical Imaging*, 20:900–916, 2001.

[62] D Cabrera Fernández, N Villate, C A Puliafito, and P J Rosenfeld. Comparing total macular volume changes measured by optical coherence tomography with retinal lesion volume estimated by active contours. *Investigative Ophthalmology Visual Science*, 45:3072, 5 2004.

[63] Ahmed Gawish, Paul Fieguth, Sebastian Marschall, and Kostadinka Bizheva. Undecimated hierarchical active contours for oct image segmentation. pages 882–886, 2014.

[64] Jelena Novosel, Gijs Thepass, Hans G Lemij, Johannes F de Boer, Koenraad A Vermeer, and Lucas J van Vliet. Loosely coupled level sets for simultaneous 3d retinal layer segmentation in optical coherence tomography. *Medical Image Analysis*, 26:146–158, 2015.

[65] Jie Wang, Miao Zhang, Alex D Pechauer, Liang Liu, Thomas S Hwang, David J Wilson, Dengwang Li, and Yali Jia. Automated volumetric segmentation of retinal fluid on optical coherence tomography. *Biomed. Opt. Express*, 7:1577–1589, 4 2016.

[66] Akshaya Mishra, Alexander Wong, Kostadinka Bizheva, and David A Clausi. Intra-retinal layer segmentation in optical coherence tomography images. *Opt. Express*, 17:23719–23728, 12 2009.

[67] Mona K Garvin, Michael D Abramoff, Randy Kardon, Stephen R Russell, Xiaodong Wu, and Milan Sonka. Intraretinal layer segmentation of macular optical coherence tomography images using optimal 3-d graph search. *IEEE Transactions on Medical Imaging*, 27:1495–1505, 2008.

[68] Michael D Abràmoff, Kyungmoo Lee, Meindert Niemeijer, Wallace L M Alward, Emily C Greenlee, Mona K Garvin, Milan Sonka, and Young H Kwon. Automated segmentation of the cup and rim from spectral domain oct of the optic nerve head. *Investigative Ophthalmology Visual Science*, 50:5778–5784, 12 2009.

[69] Stephanie J Chiu, Xiao T Li, Peter Nicholas, Cynthia A Toth, Joseph A Izatt, and Sina Farsiu. Automatic segmentation of seven retinal layers in sdoct images congruent with expert manual segmentation. *Opt. Express*, 18:19413–19428, 8 2010.

[70] Stephanie J Chiu, Joseph A Izatt, Rachelle V O'Connell, Katrina P Winter, Cynthia A Toth, and Sina Farsiu. Validated automatic segmentation of amd pathology including drusen and geographic atrophy in sd-oct images. *Investigative Ophthalmology Visual Science*, 53:53–61, 1 2012.

[71] Md. Akter Hussain, Alauddin Bhuiyan, and Kotagiri Ramamohanarao. Disc segmentation and bmo-mrw measurement from sd-oct image using graph search and tracing of three bench mark reference layers of retina. pages 4087–4091, 2015.

[72] Boglárka A N D Somfai Gábor Márk A N D Lee Wen-Hsiang A N D Smiddy William E A N D Cabrera DeBuc Delia Tian Jing and Varga. Real-time automatic segmentation of optical coherence tomography volume data of the macular region. *PLOS ONE*, 10:1–20, 2 2015.

[73] Jianxu Chen, Lin Yang, Yizhe Zhang, Mark S Alber, and Danny Z Chen. Combining fully convolutional and recurrent neural networks for 3d biomedical image segmentation. *CoRR*, abs/1609.01006, 2016.

[74] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *CoRR*, abs/1802.02611, 2018.

[75] Alfred Fuller, Robert Zawadzki, Stacey Choi, David Wiley, John Werner, and Bernd Hamann. Segmentation of three-dimensional retinal image data. *IEEE Transactions on Visualization and Computer Graphics*, 13:1719–1726, 2007.

[76] M A Mayer, R P Tornow, R Bock, J Hornegger, and F E Kruse. Automatic nerve fiber layer segmentation and geometry correction on spectral domain oct images using fuzzy c-means clustering. *Investigative Ophthalmology Visual Science*, 49:1880, 5 2008.

[77] K A Vermeer, J van der Schoot, H G Lemij, and J F de Boer. Automated segmentation by pixel classification of retinal layers in ophthalmic oct images. *Biomed. Opt. Express*, 2:1743–1756, 6 2011.

[78] Ana Carolina Lorena and André CPLF De Carvalho. Uma introdução às support vector machines. *Revista de Informática Teórica e Aplicada*, 14(2):43–67, 2007.

[79] Janmenjoy Nayak, Bighnaraj Naik, and HSr Behera. Fuzzy c-means (fcm) clustering algorithm: a decade review from 2000 to 2014. In *Computational Intelligence in*

Data Mining-Volume 2: Proceedings of the International Conference on CIDM, 20-21 December 2014, pages 133–149. Springer, 2015.

[80] Chuan Long Li, Ying Li, and Xue Rui Wu. Novel fuzzy c-means segmentation algorithm for image with the spatial neighborhoods. 2012.

[81] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36, 1980.

[82] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. volume 25. Curran Associates, Inc., 2012.

[83] Holger R Roth, Le Lu, Jiamin Liu, Jianhua Yao, Ari Seff, Kevin M Cherry, Lauren Kim, and Ronald M Summers. Improving computer-aided detection using convolutional neural networks and random view aggregation. *CoRR*, abs/1505.03046, 2015.

[84] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[85] Xiaoming Liu, Tianyu Fu, Zhifang Pan, Dong Liu, Wei Hu, Jun Liu, and Kai Zhang. Automated layer segmentation of retinal optical coherence tomography images using a deep feature enhanced structured random forests classifier. *IEEE Journal of Biomedical and Health Informatics*, 23:1404–1416, 2019.

[86] Peter Kontschieder, Samuel Rota Bulò, Horst Bischof, and Marcello Pelillo. Structured class-labels in random forests for semantic image labelling. In *2011 International Conference on Computer Vision*, pages 2190–2197, 2011.

[87] Tri Huynh, Yaozong Gao, Jiayin Kang, Li Wang, Pei Zhang, Jun Lian, and Dinggang Shen. Estimating ct image from mri data using structured random forest and auto-context model. *IEEE Transactions on Medical Imaging*, 35(1):174–183, 2016.

[88] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[89] M Pekala, N Joshi, T Y Alvin Liu, N M Bressler, D Cabrera DeBuc, and P Burlina. Deep learning based retinal oct segmentation. *Computers in Biology and Medicine*, 114:103445, 2019.

[90] Suman Sedai, Bhavna J Antony, Ravneet Rai, Katie Jones, Hiroshi Ishikawa, Joel S Schuman, Gadi Wollstein, and Rahil Garnavi. Uncertainty guided semi-supervised segmentation of retinal layers in oct images. *CoRR*, abs/2103.02083, 2021.

[91] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning, 2016.

[92] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[93] N. Man, S. Guo, K. F.C. Yiu, and C. K.S. Leung. Multi-layer segmentation of retina oct images via advanced u-net architecture. *Neurocomputing*, 515, 2023.

[94] Abhijit Guha Roy, Sailesh Conjeti, Sri Phani Krishna Karri, Debdoot Sheet, Amin Katouzian, Christian Wachinger, and Nassir Navab. Relaynet: retinal layer and fluid segmentation of macular optical coherence tomography using fully convolutional networks. *Biomed. Opt. Express*, 8:3627–3642, 8 2017.

[95] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. *CoRR*, abs/1505.04366, 2015.

[96] Omid E David and Nathan S Netanyahu. Deeppainter: Painter classification using deep convolutional autoencoders. In *Artificial Neural Networks and Machine Learning–ICANN 2016: 25th International Conference on Artificial Neural Networks, Barcelona, Spain, September 6-9, 2016, Proceedings, Part II 25*, pages 20–28. Springer, 2016.

[97] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[98] Jason Kugelman, Joseph Allman, Scott A Read, Stephen J Vincent, Janelle Tong, Michael Kalloniatis, Fred K Chen, Michael J Collins, and David Alonso-Caneiro. A comparison of deep learning u-net architectures for posterior segment oct retinal layer segmentation. *Scientific Reports*, 12:14888, 2022.

[99] Zeshan Hussain, Francisco Gimenez, Darvin Yi, and D Rubin. Differential data augmentation techniques for medical imaging classification tasks. *AMIA ... Annual Symposium proceedings. AMIA Symposium*, 2017:979–984, 2017.

[100] Phillip Chlap, Hang Min, Nym Vandenberg, Jason Dowling, Lois Holloway, and Annette Haworth. A review of medical image data augmentation techniques for deep learning applications. *Journal of Medical Imaging and Radiation Oncology*, 65:545–563, 2021.

[101] Ozan Oktay, Jo Schlemper, Lo{̈ı}c Le Folgoc, Matthew C H Lee, Mattias P Heinrich, Kazunari Misawa, Kensaku Mori, Steven G McDonagh, Nils Y Hammerla, Bernhard Kainz, Ben Glocker, and Daniel Rueckert. Attention u-net: Learning where to look for the pancreas. *CoRR*, abs/1804.03999, 2018.

[102] Tsung-Yi Lin, Priya Goyal, Ross B Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017.

[103] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[104] Xin Yi, Ekta Walia, and Paul Babyn. Generative adversarial network in medical imaging: A review. *Medical image analysis*, 58:101552, 2019.

[105] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016.

[106] Abhay Kumar Yadav, Sohil Shah, Zheng Xu, David W. Jacobs, and Tom Goldstein. Stabilizing adversarial nets with prediction methods. *CoRR*, abs/1705.07364, 2017.

[107] Ekin Tiu. Metrics to evaluate your semantic segmentation model. https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2, Oct 2020.

[108] Karan Jakhar. Dice coefficient, iou. https://karan-jakhar.medium.com/100-days-of-code-day-7-84e4918cb72c, Apr 2020.

[109] Carlos E Cardenas, Jinzhong Yang, Brian M Anderson, Laurence E Court, and Kristy B Brock. Advances in auto-segmentation. In *Seminars in radiation oncology*, volume 29, pages 185–197. Elsevier, 2019.

Appendix A

A.1 Architecture Diagrams

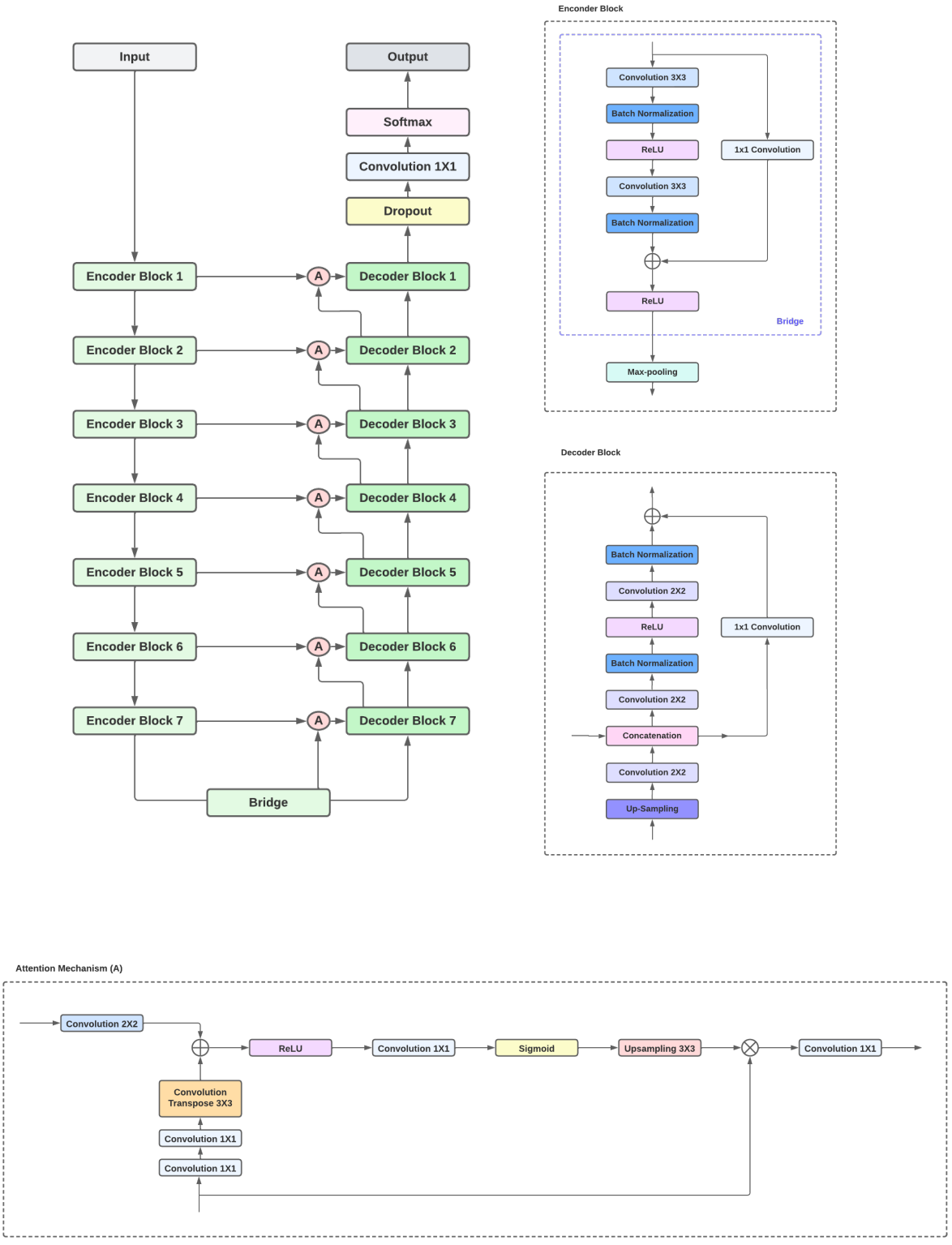


Figure A.2: Attention Res-U-Net architecture.

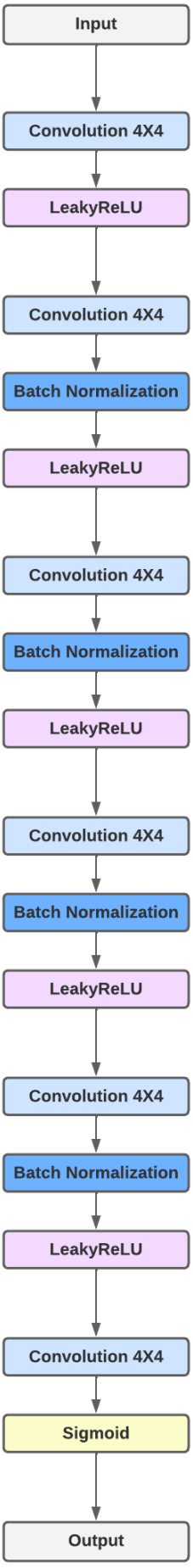


Figure A.3: PatchGAN architecture.

A.2 Monitoring Plots

Figure A.4: Loss during training and validation vs. the number of epochs for the Att-Res-U-Net.

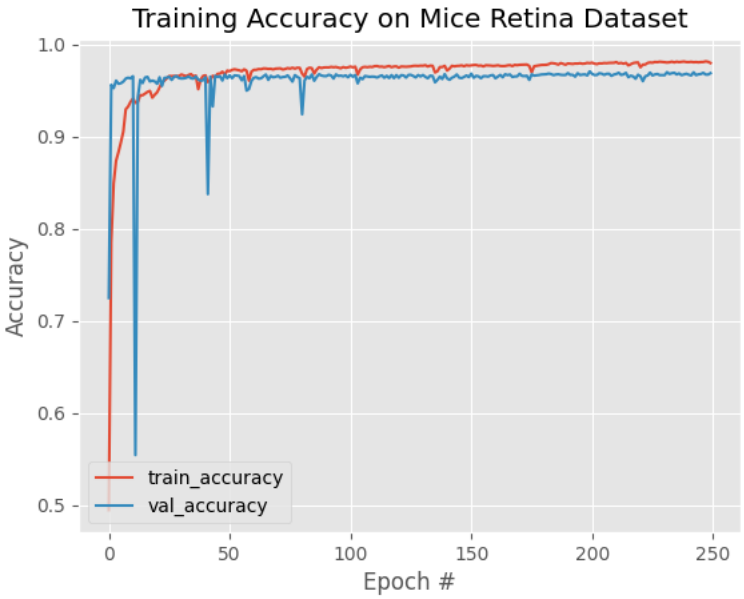


Figure A.5: Accuracy during training and validation vs. the number of epochs for the Att-Res-U-Net.

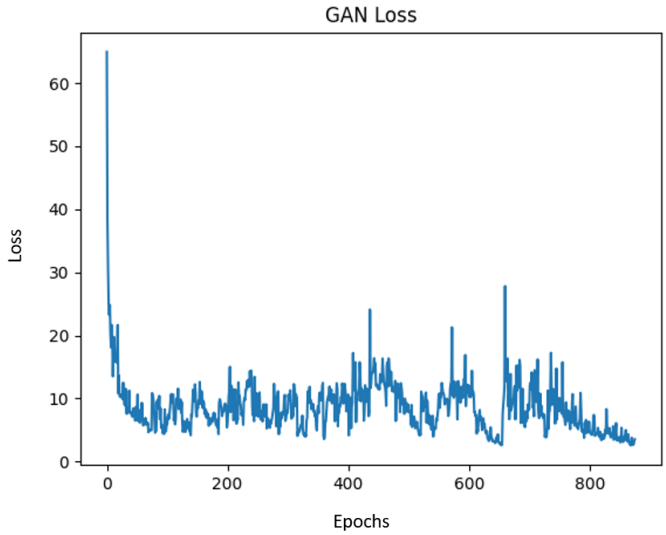
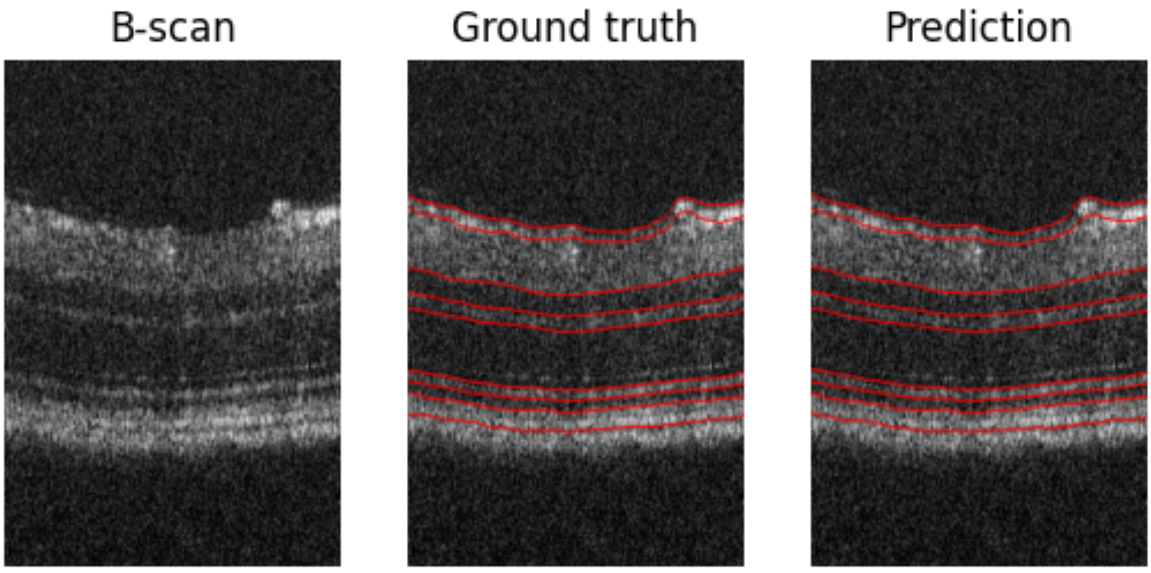


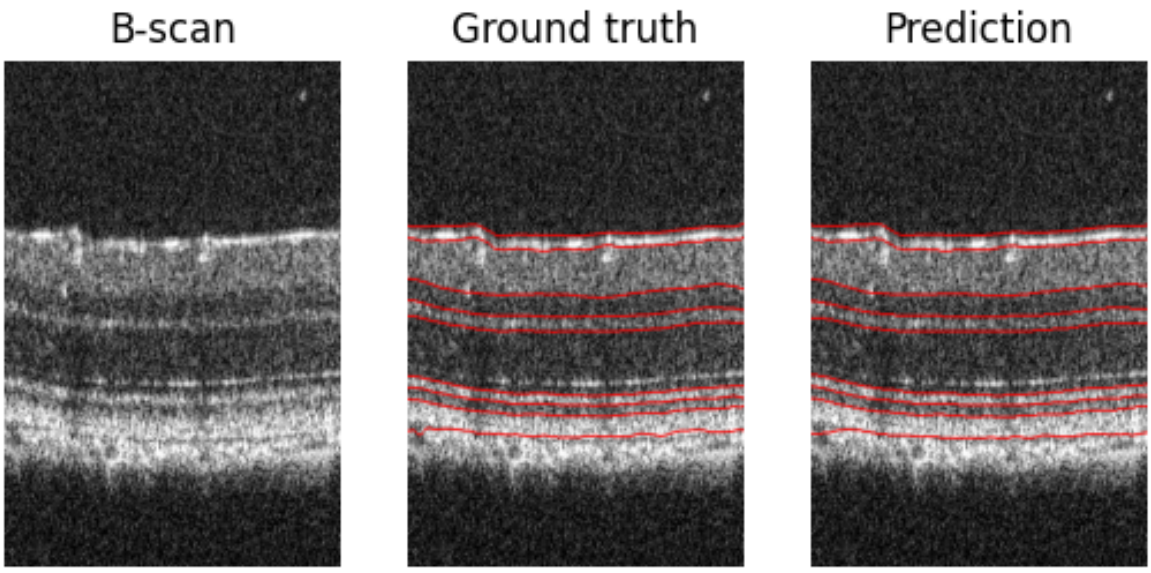
Figure A.6: Loss during training vs. the number of epochs for the Teacher-Student GAN.

A.3 Additional Segmentations

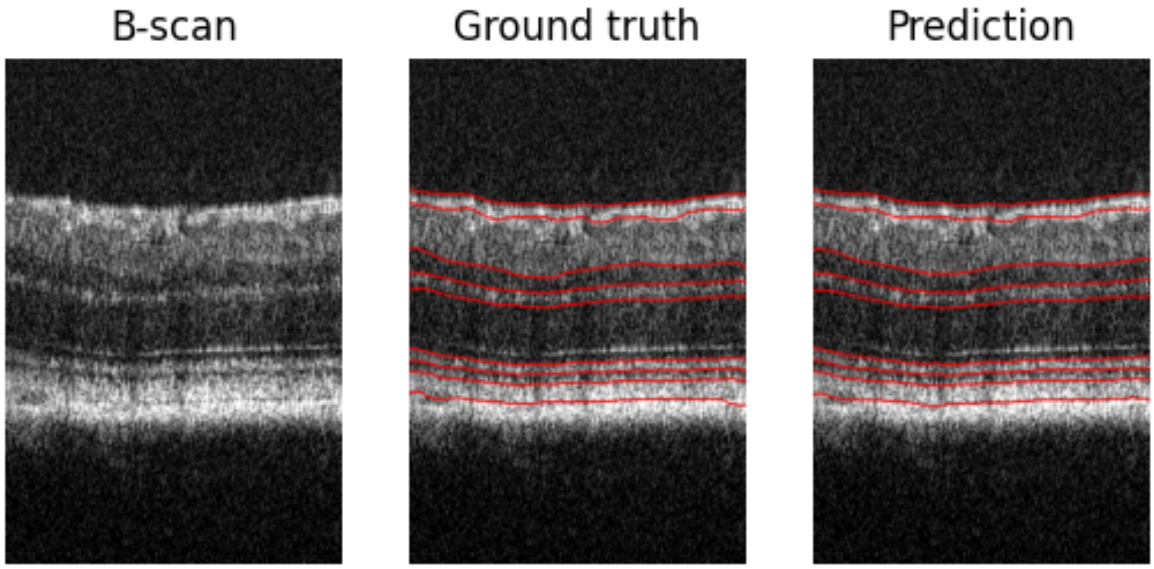
A.3.1 Dataset 1 w/ Attention Res-U-Net



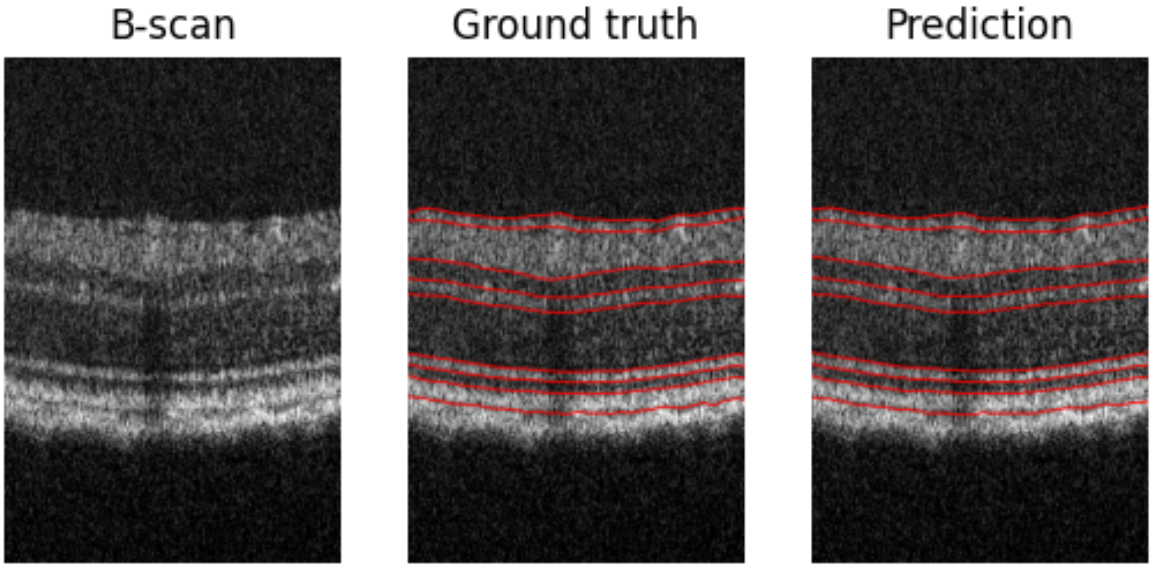
(a)



(b)



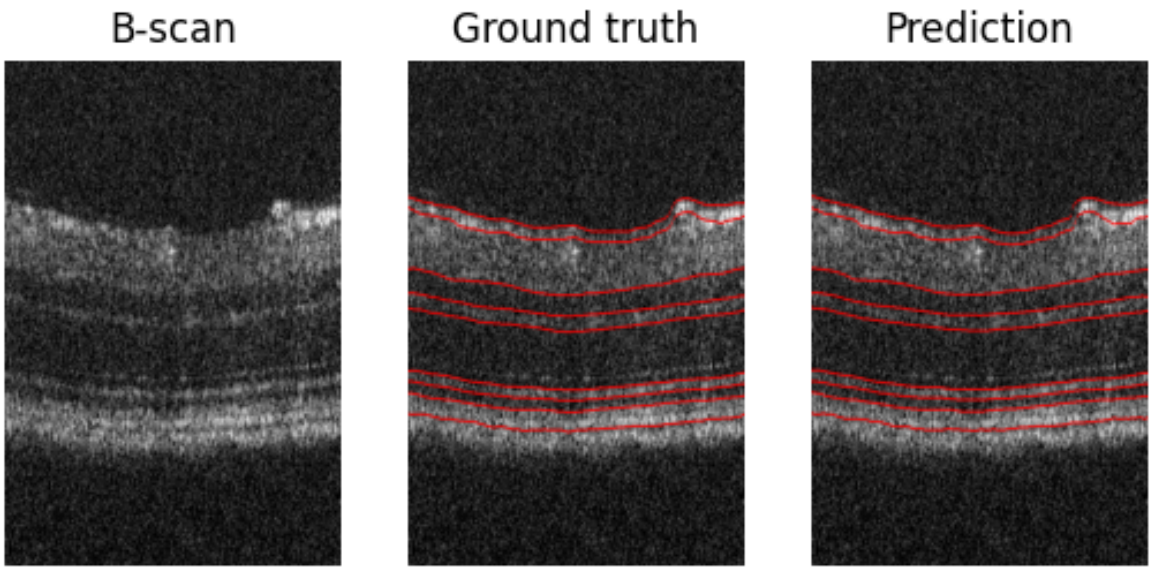
(c)



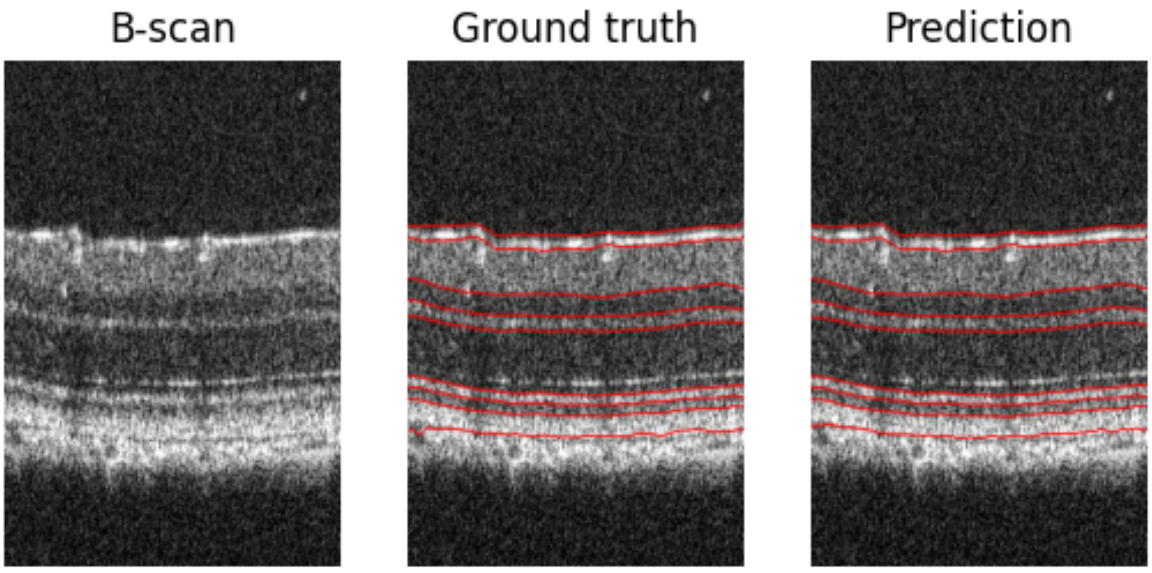
(d)

Figure A.7: Predicted layer interfaces by the Attention Res-U-Net, corresponding ground truth segmentations, and original B-scans for the DS1 test set.

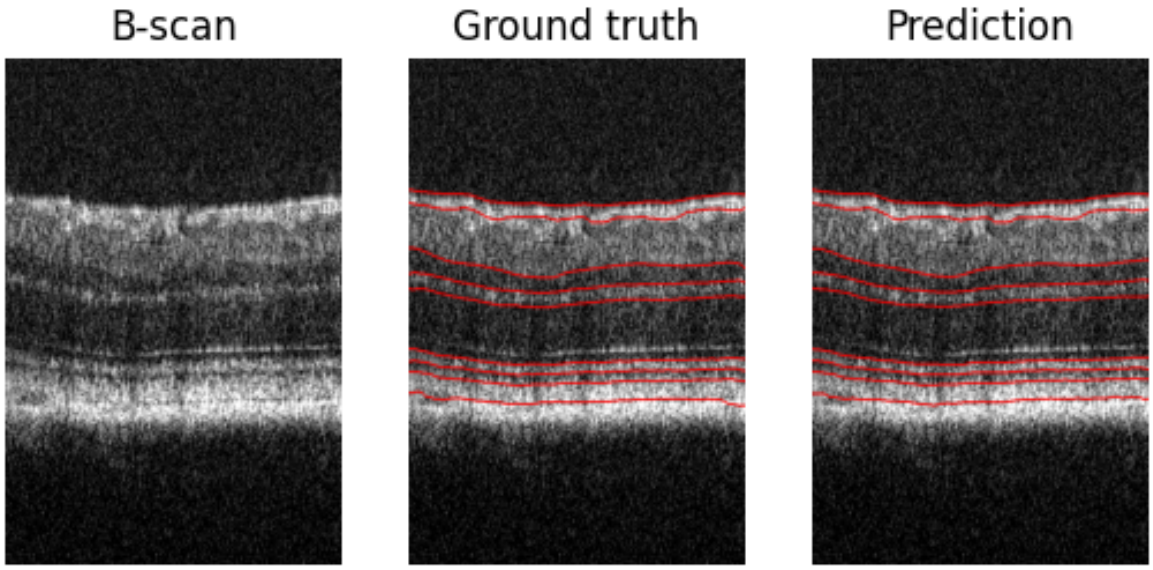
A.3.2 Dataset 1 w/ Teacher-Student GAN



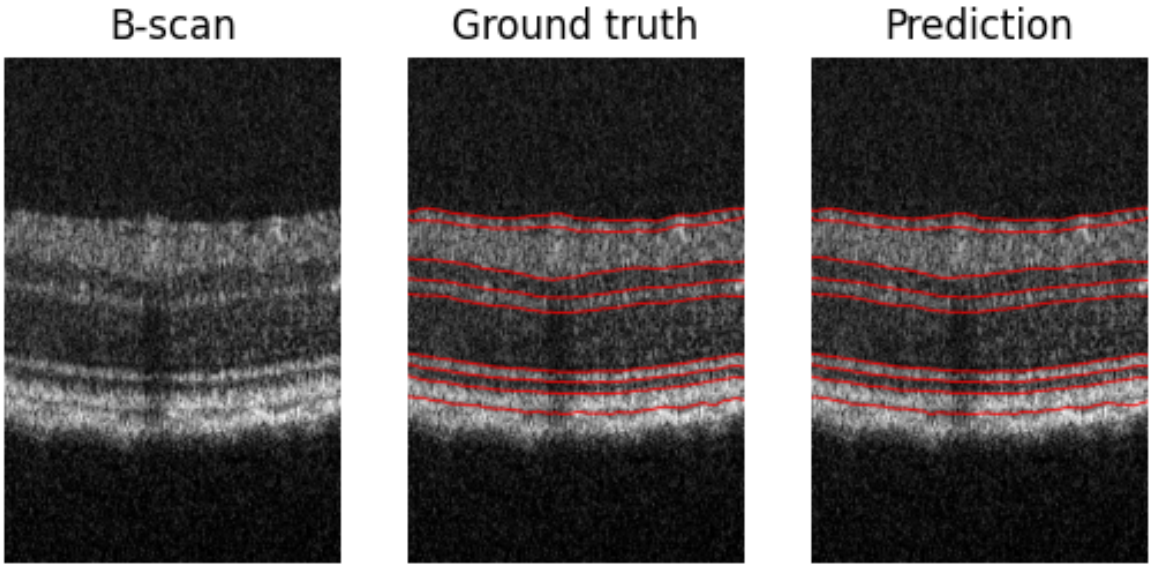
(a)



(b)



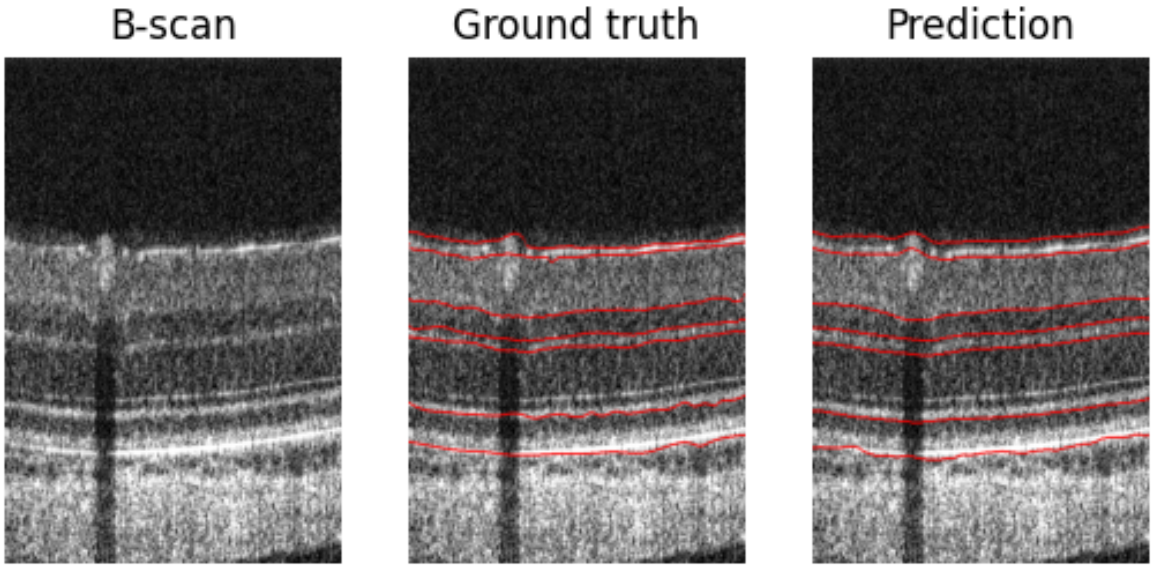
(c)



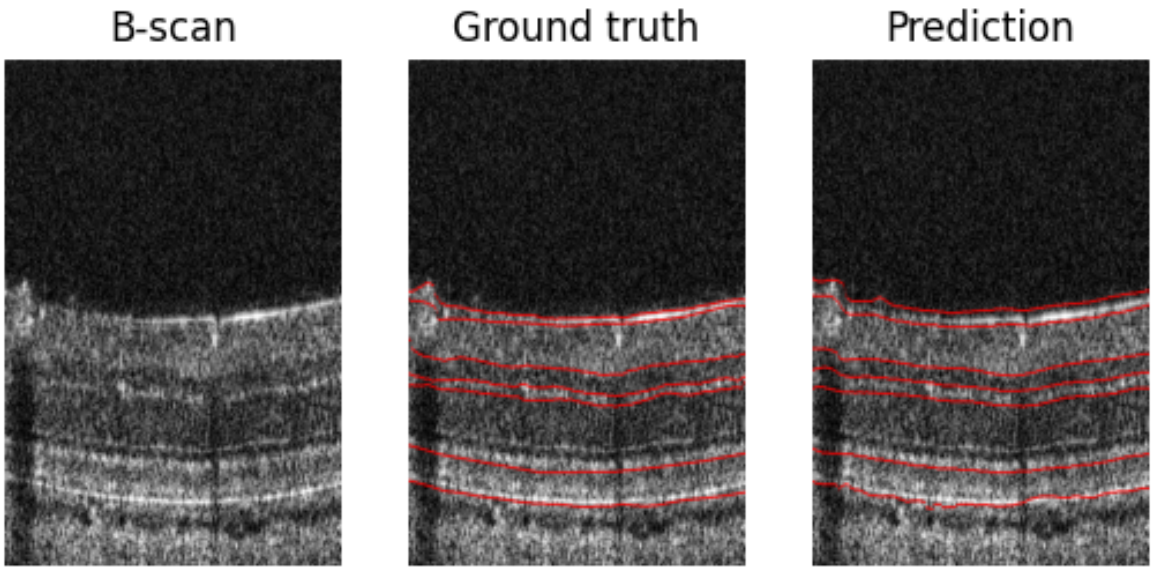
(d)

Figure A.8: Predicted layer interfaces by the Teacher-Student GAN, corresponding ground truth segmentations, and original B-scans for the DS1 test set.

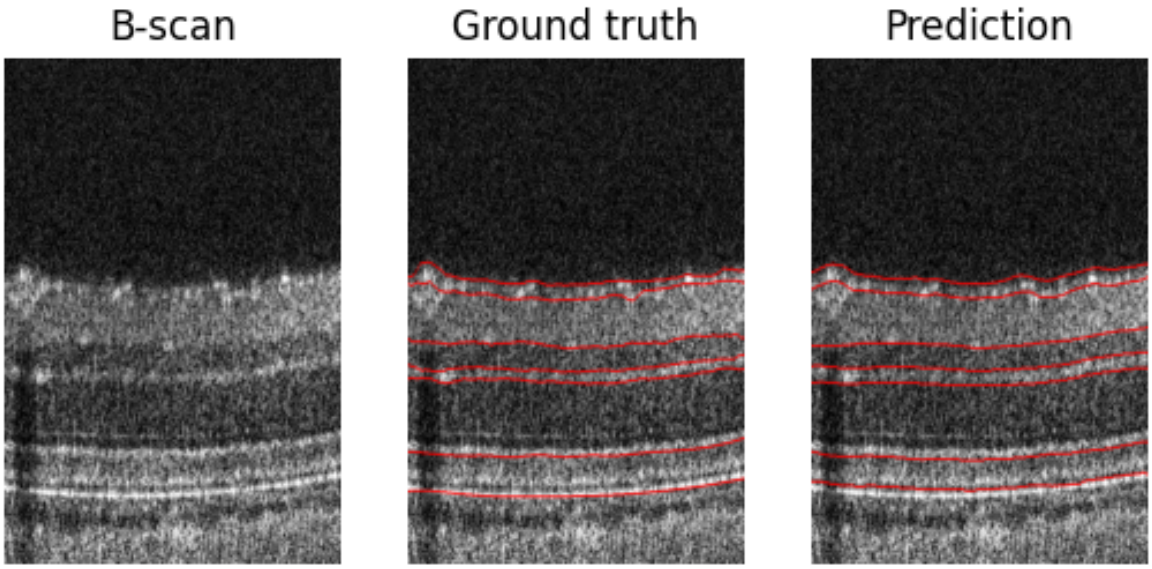
A.3.3 Dataset 2



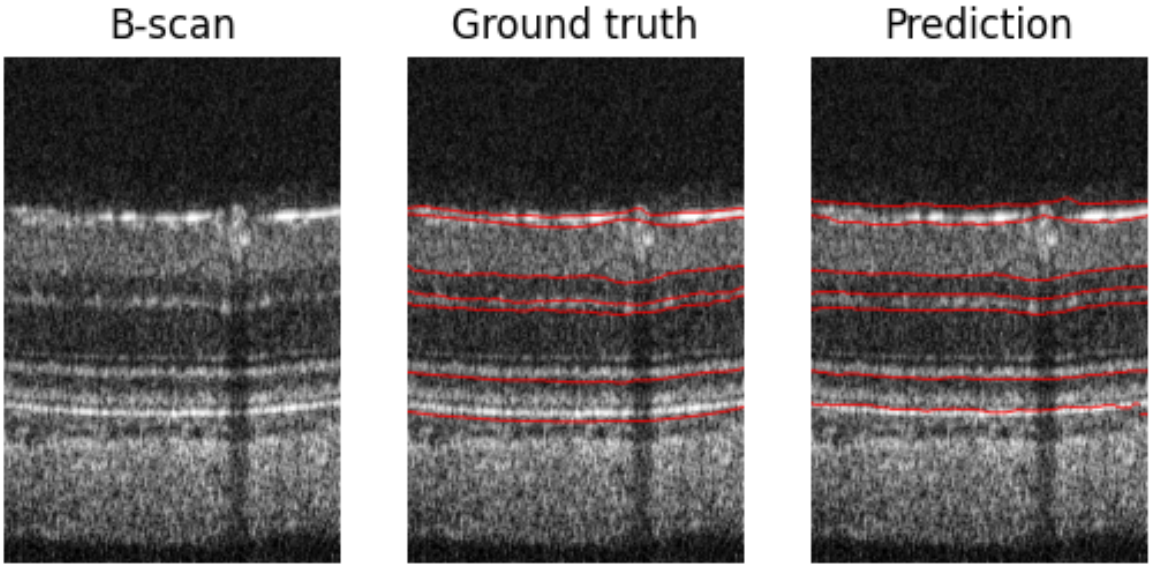
(a)



(b)



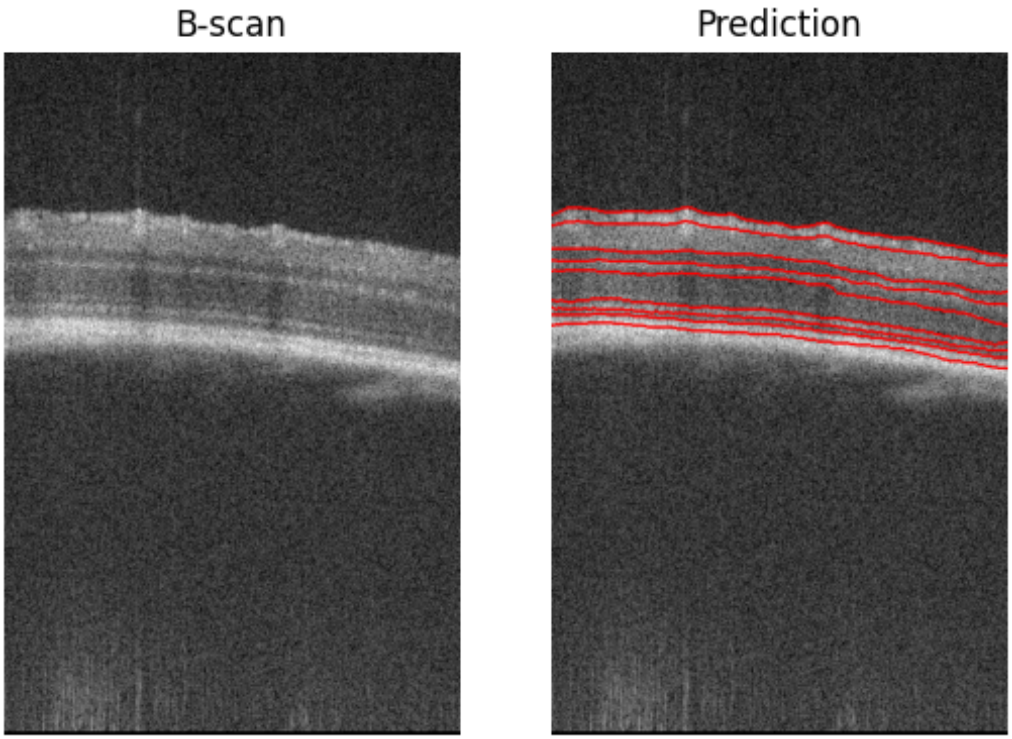
(c)



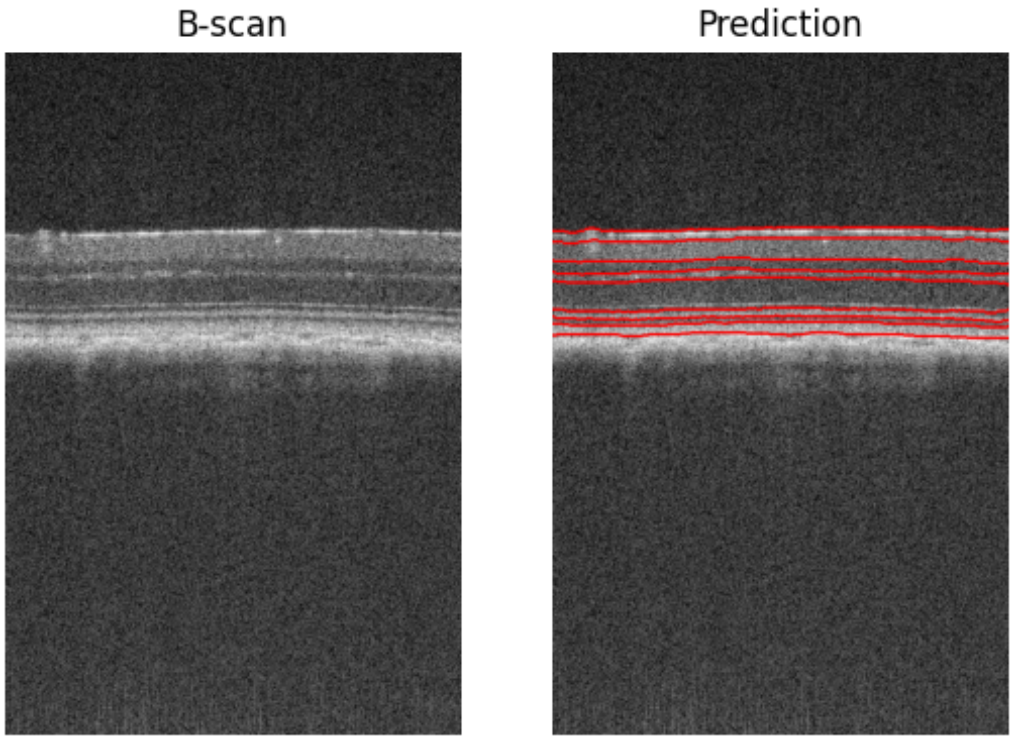
(d)

Figure A.9: Predicted layer interfaces by the Attention-Res-U-Net, corresponding ground truth segmentations, and original B-scans for the DS2 test set.

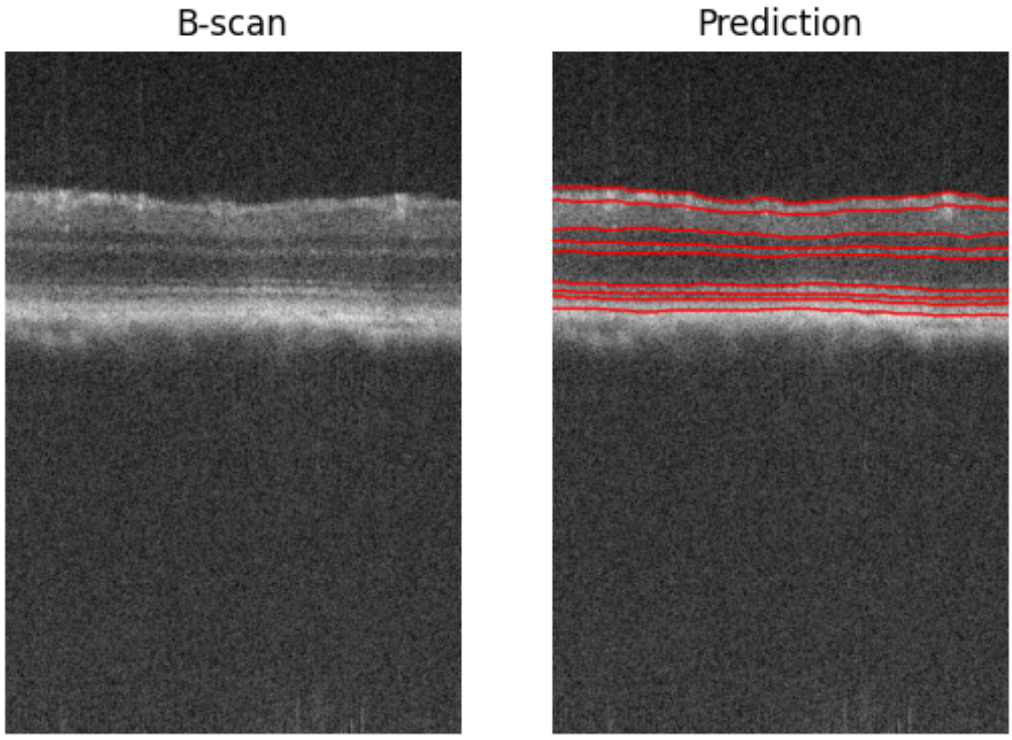
A.3.4 Dataset 3



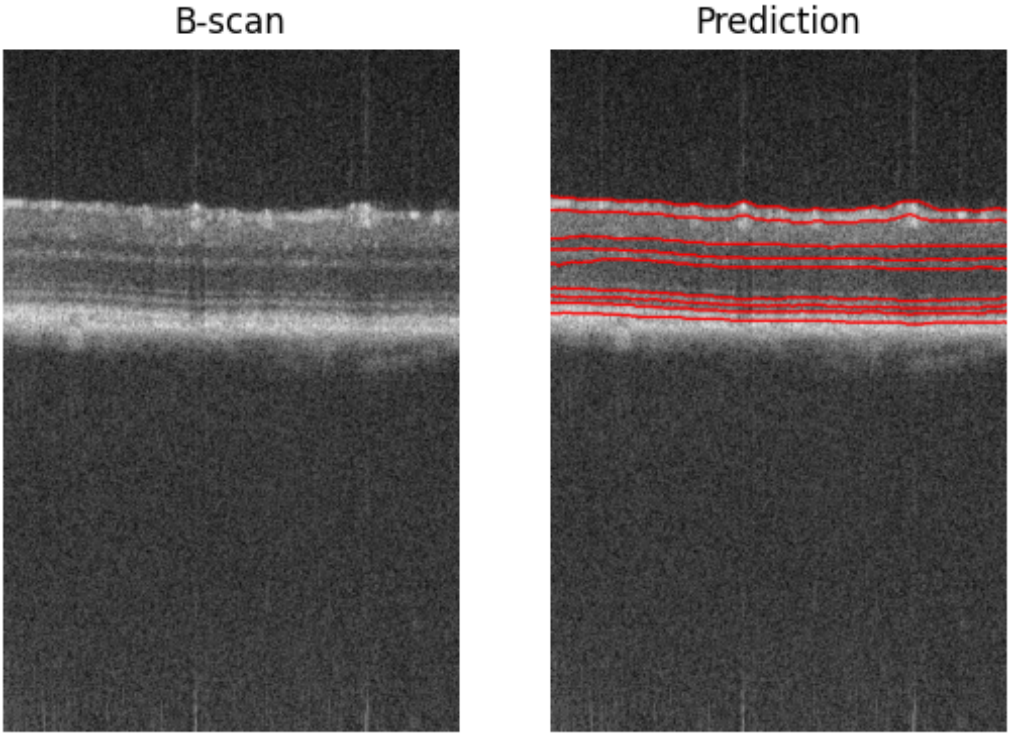
(a)



(b)



(c)



(d)

Figure A.10: Predicted layer interfaces by the Attention Res-U-Net (right) and corresponding B-scan (left) for the DS3 test set.