Deep Learning Estimation of Retinal Sensitivity from Optical Coherence Tomography in Retinitis Pigmentosa Patients

Yuxuan Li

A dissertation submitted in partial fulfillment

of the requirements for the degree of

MSc Data Science and Machine Learning

of

University College London.

Department of Computer Science

University College London

September 18, 2024

I, Yuxuan Li, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Objective: This study aims to develop a deep learning (DL) model to estimate retinal sensitivity depending on optical coherence tomography (OCT) images in patients with retinitis pigmentosa (RP), a genetic disorder characterized by progressive peripheral vision loss.

Methods: A convolutional neural network (CNN) was trained on a dataset of OCT images paired with corresponding microperimetry data, representing retinal sensitivity. The model was designed to predict the retinal sensitivity based only on structural OCT scans. The dataset included OCT images from 27 RP patients.

Results: A total of 121,551 OCT B-scan slices with their corresponding sensitivity values were generated from 82 eyes, the number of slices in each set is 95,054 in training, 12,154 in validation and 14,343 in testing. The mean absolute error (MAE) is 7.41 dB with a 95% CI from 7.36 dB to 7.46 dB when using the classic LeNet5 model. After evaluating the new network with test set, the MAE is approximately 6.53 dB with a 95% confidence interval from 6.47 dB to 6.59 dB. Correlation showed a moderate to high degree of agreement (Pearson correlation r = 0.64). By paired Wilcoxon rank sum test ($P < .001$), our model performed significantly better than the baseline model LeNet5.

Conclusions: Sensitivity maps were generated using the model-predicted sensitivity values. Compared with traditional microperimetry, the model-created sensitivity map has a higher resolution and wider coverage. The model could be used as a part

of clinical trials for pre- and after-treatment comparison.

Keywords: Deep Learning; Convolutional Neural Network; Retinitis Pigmentosa; Optical Coherence Tomography; Microperimetry; Retinal Sensitivity

Acknowledgements

I would like to thank my supervisor, Dr. Adam Dubis, for his helpful guidance and for providing essential support whenever I encountered a problem during this project. I truly appreciate the time and effort he has dedicated to helping me.

Contents

Chapter 1

Introduction

Retinitis Pigmentosa (RP) is a genetic disorder that affects the retina, it results in the progressive loss of photoreceptor cells, which makes it a leading cause of inherited blindness [1]. RP significantly impacts individuals by causing progressive vision loss and eventual blindness, with variations based on genetic transmission. Among various types of RP, the most common type sporadic RP is heterogeneous, with early-onset severe forms often indicating recessive inheritance and late-onset forms possibly linked to dominant mutations [2]. Another type X-linked RP is particularly severe, leading to blindness before age 25, often starting with myopia or night blindness [2]. The variability and severity of RP emphasize the need for advances in research and early diagnosis.

Gaining a clear understanding of how patients function visually in the context of retinal diseases is critical for guiding clinical decision-making. Traditional measures such as visual acuity are not effective indicators, since they tend to change at a very gradual speed over time as shown in research on patients with USH2A mutations [3]. Those measures often fail to capture the broader aspects of vision impairment. Patients may retain good central visual acuity, but experience a steady and significant loss in peripheral vision [4], which profoundly affect their overall ability to navigate daily life. As such, assessing visual function beyond central acuity is essential for accurately understanding disease impact and progression.

Microperimetry (MP) is one method used to assess visual function by testing the sensitivity of specific regions of the retina. It projects light stimuli of varying

intensities onto specific retinal locations and records the patient's responses, determining the threshold at which light is detected. This method provides a detailed map of retinal sensitivity, offering understanding into how different areas of the retina function, and how retinal diseases are affecting a patient's visual abilities [5].

However, MP has its challenges. It requires the patient to be attentive throughout the test, as well as adequate training for both patients and examiners to ensure reliable results. Moreover, the test can be lengthy, with each eye requiring up to 15 minutes for a thorough assessment [6]. This extended time can be exhausting for patients, and the need for human attention during testing limits the practicability of large-scale functional evaluations using MP.

On the other hand, optical coherence tomography (OCT) is a more efficient, noninvasive and commonly-used imaging technique that provides detailed structural information about the retina. OCT scanning is faster, less expensive, and requiring less effort from both patients and examiners. However, while OCT is good at mapping retinal structure, it does not directly measure visual function. This gap in functionality poses a challenge, as understanding a patient's true visual capability requires both structural and functional insights. Since humans do not have the attention capacity for large-scale functional testing like MP, creating tools that can extract retinal function directly from structural OCT images is essential. Such advancements would improve clinical efficiency and allow for broader patient assessments.

In this project, the goal is to create a deep learning model which uses OCT scans to predict retinal sensitivity. Such model would have the potential of testing if the predicted sensitivity values have been changed after treatment in a clinical trial.

Chapter 2

Literature Review and Extended Introduction

In this chapter, the specific retinal structure impacting retinal sensitivity, appropriate deep learning models and some important methods used in this paper will be discussed combining with some inspirations from other studies. It will provide strong preparations for Chapter 3 Method.

2.1 Relationship between Retinal Structure and Sensitivity

The retina is defined as the part between the inner limiting membrane (ILM) and Bruch's membrane of the eye, which is directly located underneath the retinal pigment epithelium (RPE) [7]. However, this study does not focus on the overall retinal structure; instead, more attention is given to the specific part of the retina that is relevant to visual function.

According to Toms et al. [3], the ellipsoid zone (EZ), one of the hyperreflective bands in the outer retina, appears to be significantly affected as the patient's age increases. Gill et al. [8] also concluded that the EZ width is the most sensitive biomarker of both structural and functional decline in USH2A retinopathy, where USH2A mutations are a major genetic cause of RP.

Lenassi et al. [9] discovered that in all areas where sensitivity was marked as 0 dB, the EZ was absent, indicating that those areas had an EZ thickness of 0. Such

results were consistent in both kinetic and static perimetry.

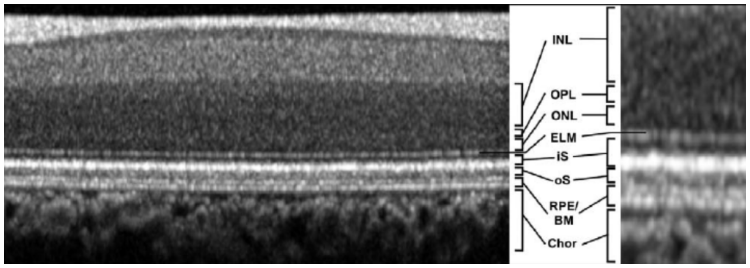


Figure 2.1: The OCT image of a healthy area at the border of the fovea centralis [10]. Abbreviations: NFL, nerve fiber layer; GCL, ganglion cell layer; IPL, innerplexiform layer; INL, inner nuclear layer; OPL, outer plexiform layer; ONL, outer nuclear layer; ELM, external limiting membrane; iS, innerphotoreceptor segments; oS, outer photoreceptor segments; RPE/BM, retinal pigment epithelium/Bruch's membrane; Chor, choroid

In Figure 2.1 [10], the EZ is the zone combining the inner photoreceptor segments (iS) and outer photoreceptor segments (oS). The EZ and other hyperreflective bands in the outer retina indicate the structural integrity of macular photoreceptors [3], which are responsible for light detection. Therefore, a reduction in the intensity and thickness of the EZ can demonstrate a deterioration in visual function.

Besides the EZ, other outer retinal layers are highly related to visual function, including the outer nuclear layer (ONL) and external limiting membrane (ELM). A study by Dubis et al. [11] suggested a strong correlation between ONL thickness and sensitivity values. Hagag et al. [12] also highlighted the connection between visual function and the peripheral loss of photoreceptor bands, including the ONL, ELM, and EZ.

Thus, it can be concluded that photoreceptors, located in the outer retina, are directly responsible for light detection. As a result, the thickness of the outer retina shows a strong correlation with retinal sensitivity, as demonstrated in another study by Kim et al. [13].

Combining all this information, the thickness this paper focuses on is the outer retinal thickness, which can be derived from the distance between the outer plexiform layer (OPL) and the inner border of the RPE (IBRPE).

2.2 Deep Learning Networks and Layers

In Inoda et al.'s study [14], the authors aimed to develop a deep learning model to estimate best-corrected visual acuity (BCVA) using cross-sectional OCT B-scans from patients with various retinal diseases. They utilized 2,700 B-scans from 756 eyes, taken in both horizontal and vertical planes, to train the model. The model architecture was based on GoogLeNet, chosen for its ability to handle complex image features. Data augmentation techniques, including horizontal flipping and random cropping, were applied to enhance the training process. The model was trained with stochastic gradient descent (SGD) and a learning rate of 1.0×10^{-7} to predict BCVA based on OCT features representing retinal structures like macular thickness. The model achieved a mean absolute error (MAE) of 0.321 on the internal test set, indicating reasonable accuracy. However, in external validation, the model's performance dropped, reflecting a challenge in generalizing across different datasets, with R^2 decreasing to 0.19. This emphasizes the difficulty of building models that generalize well to new patient populations, especially when predicting a clinically relevant metric like BCVA. Although the model they built was not designed to predict retinal sensitivity values directly, the used data type was similar to the data used in this project, their preprocessing step is worth trying.

Kihara et al. [6] used a modified Visual Geometry Group (VGG) 16 network to predict retinal sensitivity value from OCT scans collected from patients with Macular Telangiectasia Type 2 (MacTel), and their work was a great inspiration for this project. Vertical slices of size 32×496 out of B-scans were created around each MP point, and data augmentations including rotations, image translations and horizontal reflections were applied to images in the training set for overfitting reduction. The corresponding MP sensitivity values were collected as the continuous output variable of the model. The model was trained using the Adam optimizer with a batch size of 64 and mean squared error (MSE) as the loss function. The initial learning rate was set to 1×10^{-5} and decayed throughout the training process. Dropout was introduced with 0.1 after the convolutional layers and 0.5 following the fully connected layers, targeting layers with trainable weights. The pixel values of the

input images were normalized to a range between 0 and 1, while the outputs were scaled between -1 and 1. The model achieved a mean absolute error (MAE) of 3.36 dB, with a 95% confidence interval (CI) from 3.25 to 3.48 dB on the validation set and test set. Given the similarity in data size and goals, the preprocessing step of data and analysis method used in this study mostly followed the approach described in Kihara et al.'s paper, but with some modifications which would be described in detail.

In this project, a convolutional neural network will be developed, with image slices as the model's inputs. The model is designed to produce a linear output of shape 1×1. Several suitable network architectures are available for this purpose.

VGG16, used in Kihara et al.'s paper [6], was a strong candidate due to its relatively shallow architecture, making it less prone to overfitting on small datasets. Its simplicity helps avoid unnecessary complexity when dealing with limited data. However, they found that compared with shallower models like LeNet which were prone to underfitting, deeper models such as VGG16 still resulted in overfitting.

A key advantage of deeper networks like ResNet18 and ResNet34 is the inclusion of residual connections [15]. These allow the model to skip layers, improving training efficiency by addressing the vanishing gradient problem [15], which occurs when training relatively deep networks. In a residual block, also called the basic block, the input x is combined with the output of a few layers $F(x)$, where the output of the block is the sum of the original input and the output of those layers:

$$y = F(x, \{W_i\}) + x \quad [15]$$

where x is the input to the block; $F(x, \{W_i\})$ refers to the transformation applied by the layers in the block, which typically consists of a series of convolutional layers, followed by batch normalization and ReLU activation functions, with learned weights W_i; y is the final output of the block, combining both the transformed output and the original input. This residual connection ensures information to pass smoothly through the network without degrading performance, which is crucial for deep architectures.

ResNet18 is ultimately chosen over ResNet34. While both models feature residual connections, ResNet18 offers a balance between depth and simplicity. It is deep enough to capture complex patterns but not so deep as to risk overfitting or increased computational demands, making it a more suitable model for our project.

Another deep learning model GoogLeNet, used in Inoda et al.'s work [14], is not chosen because it is deeper than ResNet18 and lacks residual connections, which are vital for efficient training. Additionally, the segmentation focus of GoogLeNet adds unnecessary complexity when predicting a single sensitivity value, which drives our final decision to ResNet18.

However, ResNet18 was originally designed for the classification task instead of the regression task, it also lacks some methodologies for improving generalization and reducing overfitting, thus some modifications to ResNet18 are necessary.

A dropout layer is a common regularization technique in neural networks where a random subset of neurons is deactivated during each training iteration [16]. It prevents neurons from relying on particular features too much, forcing the network to distribute learning across multiple neurons. This random deactivation of neurons ensures that different parts of the model learn more robust and diverse features [17]. Dropout layers are particularly effective in reducing overfitting, which occurs when a model performs well on training data but struggles to generalize to unseen data [17]. By deactivating neurons randomly during training, dropout stops the network from memorizing specific patterns or noise from the training data, leading to better generalization on test or validation sets. During training, dropout acts like an ensemble method, where different subsets of the network are trained with different combinations of neurons [16]. This approach increases the model's overall performance by creating a more flexible and distributed representation of the data. Overall, dropout is an essential technique for improving the generalization ability of deep learning models, especially in cases where data is limited or overfitting is a concern. Thus, it would be beneficial to add dropout layers in ResNet18.

Another layer called the pooling layer plays a crucial role in reducing the spatial dimensions of feature maps [18], which helps manage computational complex-

ity while keeping essential information. A typical max pooling layer is used early in the ResNet18 network, where it selects the maximum value from a specified region. This process allows the network to focus on the most significant features, increasing robustness against small shifts or distortions in the input data. Pooling introduces a form of translation invariance, meaning that small changes in the input do not significantly affect the output of the network, which is beneficial for learning generalized features across different inputs. In ResNet18, pooling layers are essential for efficiently reducing the size of feature maps. This enables the network to down-sample the data while retaining relevant information, making following layers more efficient. The pooling layer also contributes to regularization, since the reduction in resolution limits the amount of detailed local information that can be passed through the layers, thus preventing overfitting of the network. Moreover, global average pooling is used in the end of ResNet18 instead of traditional fully connected layers. The global averaging pooling greatly reduces the number of trainable parameters by averaging over the spatial dimensions and reduces each feature map to a single value. This design choice minimizes overfitting risks, ensuring the network generalizes well to new data. The pooling layers, combined with the residual connections, contribute to the model's ability to learn deep representations effectively while maintaining computational efficiency and robustness.

Chapter 3

Methods

3.1 Dataset

The dataset consists of eye-scan images collected from 27 patients diagnosed with Usher syndrome with USH2A $(n = 21)$ and MYO7A $(n = 6)$ mutations [12]. Those patients were recruited from Moorfields Eye Hospital. The majority of them underwent optical coherence tomography (OCT) scans and microperimetry (MP) tests for both eyes together during two visits between 2017 and 2019, with an interval of one year between visits. Two eyes from the same patient can be treated as two separate data for OS (left eye) and OD (right eye), thus ideally the size of datasets should be 96 collected from those patients. However, there existed data from a few patients that was not complete, data from some of them did not contain images from the second visit. After sifting, a total of 82 sets of complete data were available and could be used in this project.

OCT imaging was performed using the Spectralis HRA-OCT system [19], which generated high resolution images through a volume scan of 48 B-scans displaying the structure of the retinas. The distance between each two scans was 118 micrometers. Simultaneously, a scanning laser ophthalmoscope (SLO) scan [20] was taken with each OCT scan to capture an en face view of the retina. In this procedure, SLO scans, SLO scans with locations of B-Scan lines labelled, and 49 OCT volume images corresponding to each SLO scan were all saved.

Microperimetry was carried out using the Macular Integrity Assessment

(MAIA) device [21] with Goldmann III stimuli, which have a diameter of 0.43 degrees and 4.0 mm^2 area [22]. Each test involved 37 stimuli points, including one central point, plus three concentric rings, each with radius 1°, 3° and 5°, around such point with 12 points per ring. During the MP test, retinal sensitivity was measured at each stimulus point. Similar to the OCT process, SLO technology was used during the MP test to produce retinal images as well, and the stimulus points and their corresponding sensitivity values were superimposed on these images in the MP dataset. The original SLO images with and without MP values labelled on were both saved in the MP dataset simultaneously.

Despite the images, there were also several CSV files in the dataset. The coordinates of B-scan lines of each OCT SLO were saved as CSV files.

Retinal sensitivity was measured in the decibel scale, such scale was created based on Weber's law and Fechner's law [23], and the sensitivity is proportional to the base-10 logarithm of the stimulus brightness:

$$\text{sensitivity in dB} = 10 \times log_{10}(\frac{L_{max}}{L})$$

where L_{max} indicates the maximum luminance the device can display, and L is the minimum luminance that triggers the patient's response. Based on the scale, a sensitivity of 0 dB indicates that the fraction $\frac{L_{max}}{L}$ equals 1, which means that the minimum luminance that triggers the patient's response equals the maximum luminance that the device can display, in this case, the sensitivity is poor.

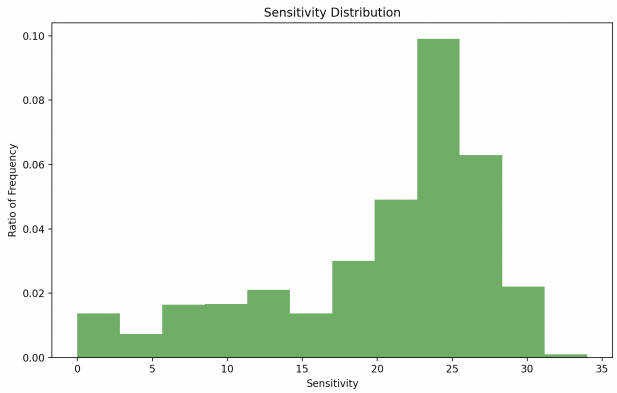
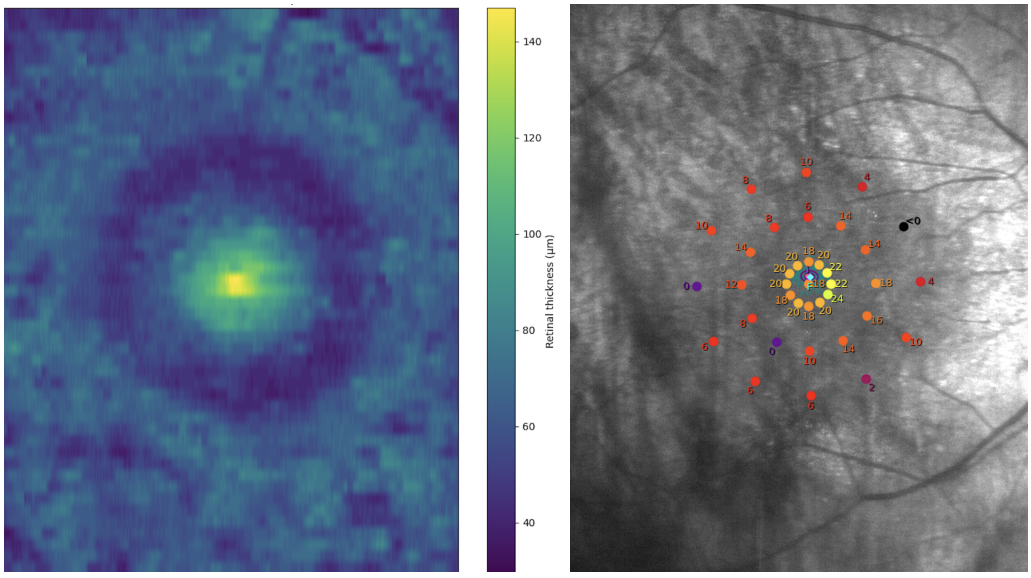


Figure 3.1: The distribution histogram of the sensitivity values

Figure 3.1 shows the distribution of sensitivity values among all MP stimuli points collected from all the MP SLOs with information labelled on. There is an obvious left skewness existing on the graph, most sensitivity values occur in the range above 20 dB, and there is little amount of MP stimuli points with sensitivity values smaller than 20 dB.

3.2 Thickness Map

The thickness map could be a direct visualization of retinal thickness, it was generated based on B-scan OCT volumes for each eye [24]. As mentioned in Section 2.1, retinal thickness used to create thickness map is the outer retinal thickness, which is the distance between the outer plexiform layer (OPL) and the inner boundary of the retinal pigment epithelium (IBRPE).



(a) (b)

Figure 3.2: (a) The thickness map of outer retina, generated from 48 B-scan OCT volumes of a patient's right eye, different colours indicate different retinal thickness. (b) The SLO image with microperimetry retinal sensitivity values labelled on of the same patient's right eye.

Each B-scan OCT volume image of size 512×512 was a cross sectional of the retina. For each column of pixels in a volume image, the thickness was calculated and recorded. After combining all the 49 volumes together, a 3-dimensional 512×49 thickness map could be created for each eye.

Figure 3.2 (a) shows a 3-dimensional outer retinal thickness map generated from the OCT images of a patient's right eye, and (b) is the corresponding MP SLO image with retinal sensitivity values labelled on. Comparing those two figures, it is noticeable that those areas with lighter colours correspond to high retinal sensitivities. Thus, it can be concluded that thicker areas of the outer retina indicate better response to light stimuli.

3.3 Data Preparation

3.3.1 OCT Image Normalization

Texture and pattern are two important features of medical images. In this study, since the model would be required to learn the image thickness, which could only be derived from the pattern, not the texture, the pattern is a more important feature over texture, and it should be prevented that the network learns too much information from image textures.

Applying min-max normalization could greatly affect the texture classification of the images [25], which would lead the network to focus more about patterns over textures.

The min-max normalization processes each pixel intensity based on the equation [26]

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

where X is the pixel intensity value, X_{max} and X_{min} indicate the maximum and minimum values of the pixel intensity of the image correspondingly, and X_{norm} is the pixel intensity after the min-max normalization. In this study, the minimum value of image intensity is 0 and the maximum value is 255 for all the images, which simplifies the equation above to

$$X_{norm} = \frac{X}{255}.$$

All OCT images were min-max normalized so that the range of pixel intensity values of these images was fixed between 0 and 1 without a specific target mean,

which would change the appearance of these images to be visually black, and efficiently help filter information to prevent the model from learning unnecessary and distracting information.

3.3.2 Image Registration

MP images were aligned to their corresponding OCT-SLO scan via a Python script implemented with the OpenCV package. The image matching process here used the article by Davidson et al. [27] as reference, modified part will be clarified.

The base SLO images without markings imprinted on top were used for registration, i.e., the pair of images using was OCT SLO without B-scan lines on, and MP SLO without stimulus points and sensitivity values on. The MP images were 1024×1024 pixels in resolution. SLO images were either 1536×1536 or 768×768 in size.

To simplify the computational complexity, both images were converted to grayscale in the Python script first. Grayscale conversion reduces the information of images to one channel, representing pixel intensity values, which was essential for the subsequent steps that rely on intensity-based feature detection. Reducing the image to a single channel maintains sufficient detail for keypoint matching while minimizing redundant data.

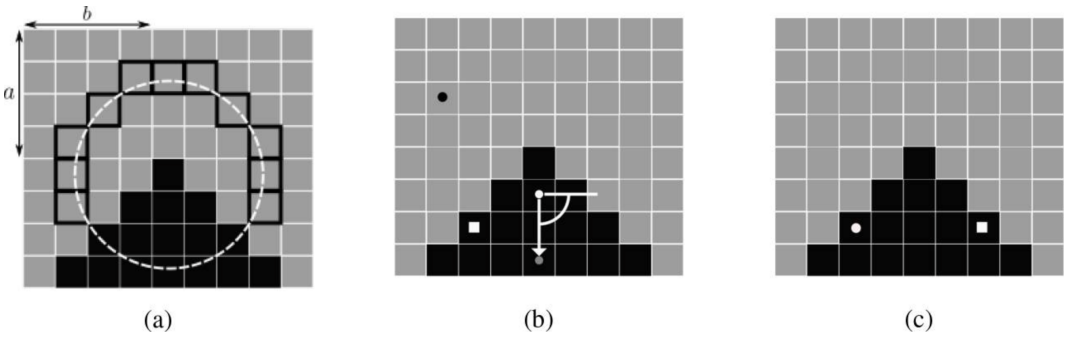


Figure 3.3: Feature detection and descriptor obtaining using ORB [27].
(a) 11 surrounding pixels (bold edges) have higher intensity than pixel (a, b).
(b) The intensity centroid (grey circle) defines the keypoint's orientation.
(c) Pixel pairs are aligned to the keypoint's orientation for descriptor calculation.

Next, the ORB (Oriented FAST and Rotated BRIEF) algorithm was applied to both images, the process is shown in Figure 3.3 [27]. ORB combines a modified ver-

sion of the FAST (Features from Accelerated Segment Test) corner detector with the BRIEF (Binary Robust Independent Elementary Features) descriptor. FAST identifies keypoints by detecting areas with significant intensity changes, often at corners of objects, as shown in (a), where 11 connecting pixels on the circular arc around the central pixel coordinate (a, b) have a higher intensity which is shown as bold-edged pixels. Once a keypoint is detected, the algorithm calculates its orientation by determining the intensity centroid, as shown in (b), where the gray circle represents the centroid. This orientation provides the keypoint with directional information, making the ORB algorithm rotation-invariant. BRIEF then extracts binary descriptors based on pixel pairs located around the keypoint. These descriptors encode local image patterns by comparing the intensity values of the pixel pairs. In (c), the pixel pairs are shown aligned according to the keypoint's orientation.

In the process, the BRIEF descriptor was computed by sampling pairs of pixels within a patch around the keypoint and comparing their intensity values. For each pair of pixels p_1 and p_2, the binary value is calculated as

$$\text{bit} = 1 \text{ if } I(p_1) < I(p_2); \text{ otherwise } 0$$

where $I(p_1)$ and $I(p_2)$ are the intensity values of the pixels p_1 and p_2, respectively. This comparison generates one bit of the descriptor. The full descriptor was formed by repeating this comparison for multiple pixel pairs, typically generating a 256-bit binary string. This descriptor encoded the local appearance of the keypoint in a way that was invariant to small changes in lighting and viewpoint.

By extracting up to 5,000 keypoints per image as suggested by Davidson et al. [27], ORB ensured that there were sufficient correspondences between the two images to establish a strong geometric relationship. The descriptors produced by ORB were binary vectors, which were then used for feature matching between the images.

Once keypoints and their corresponding descriptors had been identified in both MP SLO and OCT SLO images, the next step was to match these features. This was done using a brute-force matching technique, where each descriptor from the query

image was compared with every descriptor from the reference image. The similarity between two descriptors was measured using the Hamming distance, which was a metric that counted the number of differing bits between two binary vectors. The Hamming distance $d(a,b)$ between two descriptors a and b is given by the equation

$$d(a,b) = \sum_{i=1}^{n} (a_i \oplus b_i)$$

where a_i and b_i are the binary bits in the descriptors, and $\oplus$ represents the XOR operation. The smaller the Hamming distance, the more similar the descriptors are, indicating a stronger match between the corresponding keypoints. To ensure mutual consistency in the matches, the matcher checked whether the nearest match in the reference image was also the best match for the query image.

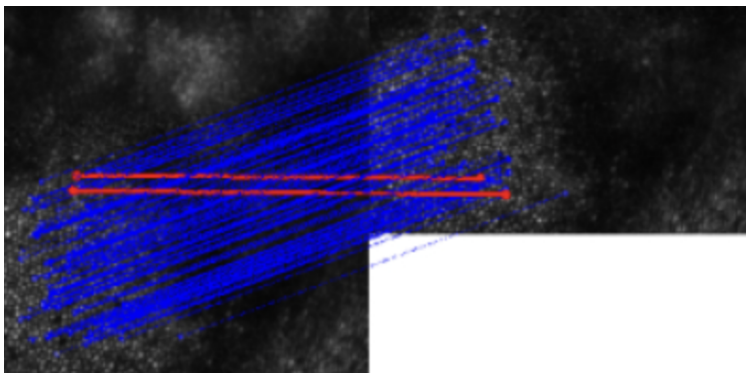


Figure 3.4: Keypoints in each image and their corresponding matches that are represented by lines [27]. There are two incorrect matches in red, which would be filtered out based on their slopes.

After finding initial matches, the matches were further refined. The method filtered out inconsistent matches by evaluating the slopes of the lines connecting the matched keypoints in both images, which was different from the methodology Davidson et al. proposed. As shown in Figure 3.4, those incorrect matches represented by red lines would be removed using the match filtration based on slope as described below.

The slope between two points was calculated using the equation

$$\text{slope}(\text{Point 1, Point 2}) = \frac{y_2 - y_1}{x_2 - x_1}$$

where (x_1, y_1) and (x_2, y_2) were the coordinates of the matched keypoints from the MP SLO and the OCT SLO images, respectively. Once the slopes were calculated, the statistical distribution of the slopes was analyzed by calculating the interquartile range (IQR), which was defined as the difference between the third quartile (Q3) and the first quartile (Q1)

$$\text{IQR} = Q3 - Q1.$$

A threshold was then applied to filter the slopes based on their deviation from the median slope. The filtering criteria were defined by the equations

$$\text{Lower Bound} = \text{Median Slope} - \text{Threshold Factor} \times \text{IQR}$$

$$\text{Upper Bound} = \text{Median Slope} + \text{Threshold Factor} \times \text{IQR}$$

where the threshold factor, the factor adjusting the range of acceptable slopes, is set to 1, 1.5 and 2 for comparison. Matches with slopes that fell outside this range $[\text{Lower Bound}, \text{Upper Bound}]$ were considered outliers and removed. This ensured that only geometrically consistent matches were saved.

After filtering the matches, an affine transformation that aligned the MP SLO with the corresponding reference image OCT SLO was estimated. An affine transformation was a linear mapping that preserved points, straight lines and parallel lines, but involved translation, rotation, scaling and shearing. The affine transformation matrix, T, could be written as

$$T = \begin{bmatrix} a & b & t_x \\ c & d & t_y \end{bmatrix}$$

where the coefficients a, b, c, d account for rotation, scaling, and shearing, and t_x, t_y represent translation. After applying the transformation to (x, y), a new coordinate (x', y') could be generated with equations

$$x' = a \cdot x + b \cdot y + t_x$$

$$y' = c \cdot x + d \cdot y + t_y$$

The transformation matrix was computed by minimizing the least-squares error between the corresponding keypoints in the MP SLO and the OCT SLO image. This process ensured that the transformation best aligned the sets of matched points. The affine transformation could thus map the MP SLO image's coordinates to the OCT SLO image's coordinates, allowing the MP SLO to be spatially aligned with the OCT SLO.

Once the affine transformation matrix had been computed, the MP SLO image was transformed using this matrix. This operation was typically performed by a function that applied the transformation matrix to every pixel in the MP SLO using the formula above. The result was a spatially transformed version of the MP SLO image, which aligned with the OCT SLO image. The output image would have the same size as the OCT SLO image, ensuring that the two images could be directly overlaid.

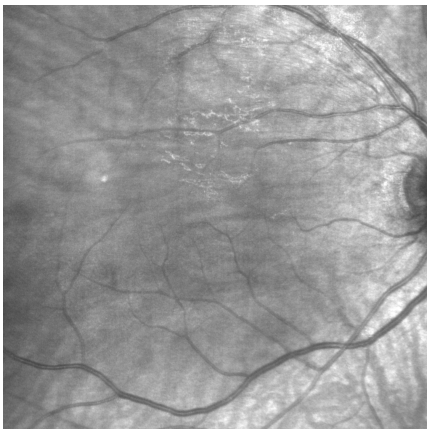


Figure 3.5: An example of an overlaid image obtaining from successful matching. Blood vessels from MP SLO and OCT SLO are exactly overlapped.

After the script was run for the first time, the next step was to manually evaluate the success of each registration. To verify the accuracy of the registration, the aligned MP SLO images were combined again with the OCT SLO images, using an overlaying technique. This allowed both images to be viewed simultaneously, providing visual confirmation that the blood vessels and other anatomical structures were correctly aligned. Successful matching was defined based on whether

the blood vessels in the MP SLO image and OCT SLO image were overlapped in the overlay image as shown in Figure 3.5. If the vessels overlapped precisely, the matching was considered successful. Only the MP SLO images that showed successful matchings were saved, while unsuccessful matchings and their corresponding files were removed. This step ensured that only the accurately aligned MP SLO images proceeded to the next stage of processing.

For cases where automatic registration fails, manual adjustments were made using Adobe Photoshop. The MP SLO images were manually rotated, cropped, or shifted to match the position of the OCT SLO images until the the blood vessels were superimposed as closely as possible. Once the manual adjustments were complete, the correctly aligned MP SLO images were saved in the same folder as the automatically succesfully-registered images. These manually adjusted MP SLO images were then used in the second run of the script.

The script was then run a second time, but this time using the subset of successfully registered MP SLO images from the first run, combined with the manually adjusted MP SLO images. Since these images had already been aligned with their corresponding OCT SLO images either automatically or manually, the matching process was expected to be fully successful. The affine transformation matrices, overlays, and registered MP SLO images were generated and saved.

This image registration process used both automated and manual alignment techniques to guarantee that all MP SLO images were accurately registered and matched the position of OCT SLO images, affine transformations were saved for further analysis.

3.3.3 MP Points Transformation and Data Standardization

From each original MP image with stimuli information labels, each central pixel coordinate of the stimuli spot at the centre was manually found and recorded to a CSV file. After that, a Python script was run to generate the coordinates of 37 transformed MP stimuli points per MP image.

The script initiated by importing the CSV file consisting of recorded central pixel coordinates. It calculated a grid of 37 coordinates around each center based

on a pixel-to-degree ratio, specifically at 0.0356, applying the conversion of degrees to pixels using the formula pixels $= \frac{1}{0.0356} \times$ degrees. For each center, the grid was adjusted to account for the eye's orientation and a specified rotation was applied in order to match the points of the second visit to the points of the first visit, which was caused by slight tilts of heads during tests. Those calculated 37 coordinates per image were of original MP images by now, equivalently, they were stimuli point coordinates of MP images that had not been registered yet.

Next, the transformation matrices derived for matching MP SLO images to OCT SLO images in the image registration process (Section 3.3.2) were used to transfer those coordinates of MP stimuli points. Thus, the transformed MP coordinates for each eye could be derived and saved for further analysis.

As mentioned in the dataset description Section 3.1, there were 2 sizes of OCT images, 1536×1536 and 768×768. Continuing to use those different sizes of images would eventually result in a chaos over data, thus data standardization was necessary.

In order to keep those dimensions the same for future convenience, SLO images with smaller sizes should all be scaled up to match the larger images. However, since the information needed from OCT SLOs had already been passed to coordinates of B-scan lines in each SLO, registered MP SLO images and cross-sectional OCT images, these OCT SLO images would not be used again afterwards, it was acceptable to only change the data mentioned above that received relating information from those images.

First, coordinates of B-scan lines from each OCT SLO image should all be standardized to match the SLO size of 1536×1536. Pixel coordinate of each line was recorded as x and y coordinates of start point and end point, which was in the form of

$$(X_{start\ point},\ X_{end\ point},\ Y_{start\ point},\ Y_{end\ point})$$

When the width and height of an image were doubled from their original width and height, simultaneously, each pixel in the original image would have the previously doubled x and y coordinate values. Thus, to scale up B-scan lines coordinates with

sizes of 768×768, x-coordinates and y-coordinates recorded of start points and end points, i.e., $X_{start\ point}$, $X_{end\ point}$, $Y_{start\ point}$ and $Y_{end\ point}$, were all multiplied by a factor of 2.

Those registered MP SLOs with two sizes 1536×1536 and 768×768 would have the only size 1536×1536 after scaling up. Similar to OCT SLOs, information needed for MP SLOs had been transferred to those MP stimuli coordinates. Same to the case of B-scan lines coordinates, both of x-coordinates and y-coordinates of MP stimuli points were multiplied by two.

Correspondingly, the sizes of cross-sectional OCT images were 1024×496 or 512×496 pixels, and smaller-sized OCT images should be resized to the larger size 1024×496. The Python Pillow library was used for this task to resize the widths of those OCT images.

After this series of scaling up actions, it could be ensured that data was standardized, all of SLOs were of size 1536×1536 pixels, coordinates matched the scale of SLO images, and all the OCT images were in the size of 1024×496.

3.3.4 Data Splitting and OCT B-Scan Slices Generation

Typically, data from the same patient cannot be partitioned into training, validation or testing datasets at the same time in order to prevent data leakage, which is caused by the contamination made by training data to the test set [28]. The existence of data leakage could diminish the model generalizability a lot by allowing the model to learn a part of test data during training [29], which is the reason of not assigning data from the same patient to different datasets. However, in this project, the amount of data was not enough to fulfill the above requirement, thus it was only ensured that data generated from the same OCT volume image was split into the same dataset.

The size of the dataset here was not large, so the model needed a large ratio of training data. Thus data was designed to be split into 80% training, 10% validation and 10% testing datasets.

It was assumed that every OCT image produced similar amount of slices. Then, all the OCT volume images were randomly split into 80 / 10 / 10 for training / validation / testing datasets. In the following slicing process, generated B-scan

slices were assigned to different datasets depending on the categories of their OCT images. This process was not able to guarantee a perfect 80 / 10 / 10 ratio, but it would generate some ratios similar to that, and it could prevent the model from learning patient-specific features which could result in overfitting.

The diameter of each Goldmann MP stimuli point is 0.43 deg, which equals 12.0787 px after converting it to pixels in MP images. After aligning MP images to OCT SLOs, the size of each MP image (1024×1024) was changed to match OCT SLOs (1536×1536), thus the diameter of each MP point became 18.11805 pixels, which rounded to 18.12 pixels. As MP stimuli points were circular areas instead of singular points, there would be areas where the points overlapped with B-scans.

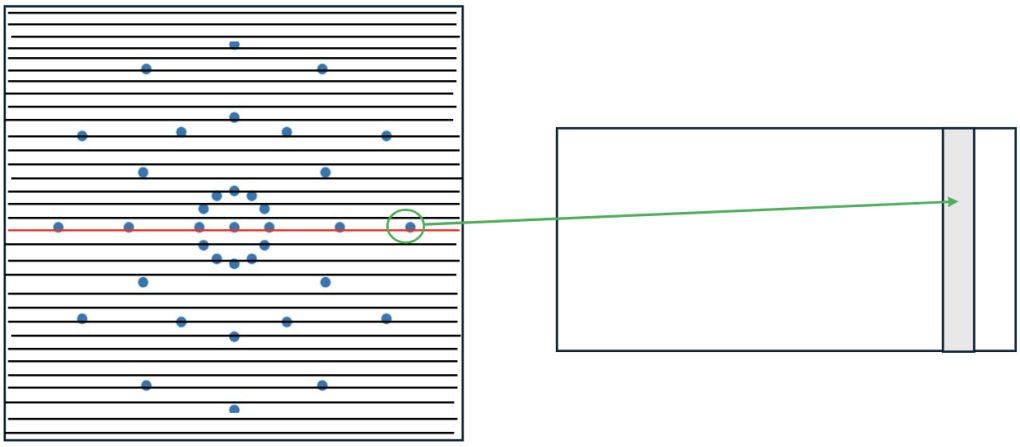


Figure 3.6: A hand-drawn sketch map of B-Scan slicing, the plot on the left should have 49 horizontal lines, but since the plot was manually drawn, the number of horizontal lines was reduced to ensure clarity.
The picture on the left shows an overlay image of registered MP SLO and OCT SLO with information labelled on, including B-scan lines from the OCT SLO and the MP stimuli points from the MP SLO. The right picture is the OCT B-scan volume image corresponding to the highlighted B-scan line on the left. The circled MP stimuli point matches the vertical slice on the right.

The B-scan slicing process was completed using a Python script, it was designed to find where the MP stimuli points overlap with B-scan lines, and cut a small vertical slice of size 32×496 around the overlapping area from the related OCT volume image of size 1024×496 as shown in Figure 3.6. These vertical windows corresponded to the regions of the retina where MP stimuli data were collected, and they were inputs for the network.

The center coordinates of 37 points per MP map were available so far, and each point was actually a circular area with around 18.12 pixels in diameter as derived above, so a circular neighborhood could be drawn around the center coordinates of each stimuli point, which had a radius of 9.06 pixels. In this neighbourhood, 261 point coordinates were selected and recorded, and then these point coordinates would be used to check if they were close enough to any of the B-scan lines. If those point were close enough to a B-scan line, it could be determined that the MP stimuli point overlapped with the line.

The method of determination on overlapping used here was cross product, whose equation is

$$\textit{Cross Product}\ (A,B,C) = (Y_C - Y_A) \times (X_B - X_A) - (X_C - X_A) \times (Y_B - Y_A)$$

where (X_A, Y_A), (X_B, Y_B) and (X_C, Y_C) are coordinates of the points A, B and C respectively. The absolute value of *Cross Product* (*start, end, point*) indicates the direct line distance between the point and the B-scan line, where *start* and *end* were start coordinate and end coordinate of a B-scan line, and the point coordinate *point* used here was the coordinate of a point within the generated circle neighbourhood of MP coordinate.

All the 261 point coordinates were looped over to check if they were close to any B-scan lines. If the absolute value of cross product was smaller than 2048, it could be considered as close to the line. Equivalently, the direct line distance between the coordinate and the B-scan line was less than 2 pixels, which was set for the existence of human errors.

If a point coordinate was considered as close to a B-scan line, the direct line distance between such coordinate and start coordinate of the line was calculated by Pythagorean Theorem, whose equation is

$$\text{hypotenuse} = \sqrt{(\text{width})^2 + (\text{height})^2}$$

Since the point coordinate was extremely close to the B-scan line, the angle formed

by the B-scan line and the line from the start point to the point coordinate is negligible, then the length of hypotenuse could be considered as equivalent to the distance between the start point and the projection of point coordinate. Given the length between the start coordinate and the projection of point coordinate, the central point position of the sliced window was obtained. A small vertical window of size 32×496 could then be sliced out, which covered the area between 16 pixels to the left from the center point and 16 pixels to the right from the point. Those slices were categorized into different datasets according to the OCT volume image they were sliced from. If the volume image was categorized as a training image, then all the slices generated from such image were training data.

As an MP stimuli point generated numerous point coordinates, and each coordinate had the potential to be close to a B-scan line, one single MP stimuli point could produce a cluster of slices, which greatly increased the data size, but raised the probability of overfitting at the same time.

3.4 Convolutional Neural Network Model

The deep learning model used in this project is a modified ResNet18, called newResNet18, which is a Convolutional Neural Network (CNN) model, as shown in Figure 3.7.

The newResNet18 model used here was constructed based on the original ResNet18 with a few modifications. The first modification involves adapting the initial convolutional layer to accommodate grayscale images by replacing the original 7×7 kernel, designed for RGB images, with a 3×3 convolution that accepts a single input channel only. The original ResNet18 model has 4 layers with each layer containing 2 basic blocks; instead, the newResNet18 model removes one of the layers, which makes the model have a shallower structure with only 6 basic blocks. Another key modification is the inclusion of dropout layers after each two basic residual block, as well as before the final fully connected layer. This addition is intended to reduce overfitting, especially when working with smaller datasets [16] [17]. The fully connected layer is also modified to output a single value instead of

multiple classes, making it suitable for regression tasks rather than classification.

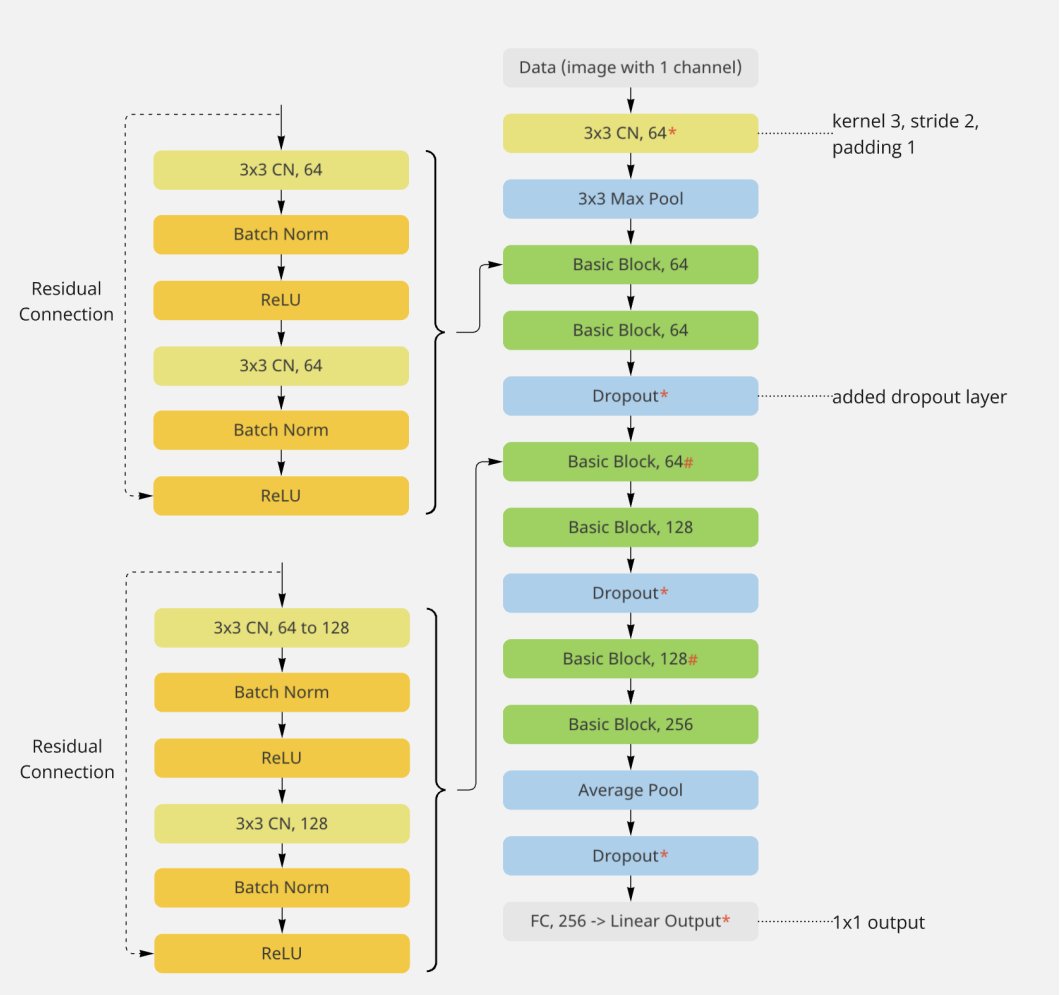


Figure 3.7: This is the architecture of the modified ResNet18, newResNet18, used in this study, it only receives grayscale images with 1 channel as inputs.
The * sign indicates that this part is different from the original ResNet18 model. the # sign indicates that this is a residual block including the down-sampling process, which increases the number of filters by applying a convolution on the input. The expanded structures within such blocks are listed near *Basic Block, 64#* as examples, and the structure of a normal block without down-sampling is listed near *Basic Block, 64* as an example.

In addition, the residual connection is the key component of ResNet18, also the reason of selecting this model, the key formula of residual connection [15] is

$$y = F(x, \{W_i\}) + x$$

where x is the input to the block; $F(x, \{W_i\})$ is the transformation applied by the

layers in the block with learned weights W_i; y is the final output of the block, combining both the transformed output and the original input. The residual connection in each basic block ensures information to be passed smoothly without degrading performance.

The newResNet18 model was trained with Adam optimizer and using mean squared error (MSE) as the loss function, which were both used in Kihara et al.'s project [6].

The MSE measures the average squared difference between the predicted values and the true labels, its formula is given by

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

where $\hat{y}_i$ is the i-th predicted value, y_i is the corresponding true observed value and n is the batch size. The MSE squares the error terms, so it penalizes large errors more than small ones.

The Adam optimizer is a first-order gradient-based optimization algorithm that uses adaptive estimates of lower-order moments for the optimization of stochastic objective functions [30]. During the training phase, the model iteratively learned by processing batches of input data from the training set. For each batch, the model made predictions and computed the training loss using MSE. The Adam optimizer then calculated the gradients of the loss with respect to the model's parameters, including weights and biases, using adaptive moment estimation. Adam maintained two moving averages [30]: the first for the gradient, referred to as the first moment, and the second for the squared gradient, known as the second moment. These moving averages allowed Adam to adjust the learning rate dynamically for each parameter, helping faster and more stable convergence. The optimizer, using these adaptive learning rates and momentum, guided the updates of the parameters, moving them in the direction that minimized the loss. This process was repeated across multiple batches and epochs, gradually refining the model's parameters to improve prediction accuracy. Throughout the training phase, the model continuously adjusted its internal parameters based on the training loss, with Adam optimizing the

step size at each update to effectively capture the relationships in the training data.

In this project, LeNet5 was selected as a baseline model due to its simplicity and historical importance in image classification tasks. The original architecture of such model had been modified to accept grayscale images with a single channel. LeNet5 comprised two convolutional layers, followed by pooling layers and fully connected layers, making it computationally efficient and less prone to overfitting in small datasets. In addition, Adam optimizer would be used in LeNet5 as well. The design of LeNet5 allows for a clear performance comparison with the more complex newResNet18, highlighting the benefits of deeper architectures with residual connections and their ability to capture more complex image representations.

3.5 Data Augmentation

During the training process, the OCT B-scan slices were first processed by a series of data augmentation steps. Initially, it was double checked that slices were already converted to grayscale with a single output channel only and with a shape of 496 pixels in height and 32 pixels in width.

After that, the slices were normalized using a mean of 0.1209 and a standard deviation of 0.1118, those two values were obtained from looping over all slices in the training set and calculating the mean and standard deviation of them. This normalization ensured that pixel intensity values were scaled appropriately.

The heavy left skewness in Figure 3.1 indicated that there were much more high sensitivity values than low sensitivity values in the datasets, thus a heavy data augmentation would be applied to those OCT slices with low sensitivity values. In order to maintain the original distribution unchanged as much as possible, a light data augmentation was applied to high frequency data, and a heavy data augmentation set was applied to low frequency data, both augmentation techniques were used with the same probability.

With a probability of 75%, a random horizontal flipping and/or a random rotation within a range of 5 counter-clockwise degrees to 5 clockwise degrees was applied as the light data augmentation. The light augmentation was applied to

specifically OCT slices with sensitivity values greater than or equal to 20 dB.

Stronger augmentations were applied to those OCT slices with sensitivity values smaller than 20 dB, including random rotations within a larger range of 8 counter-clockwise degrees to 8 clockwise degrees and/or a random horizontal flipping. These heavy augmentations were applied with a probability of 75% to those OCT slices with low-frequency sensitivity values.

Both of the basic and heavy data augmentations were not applied to OCT slices in the validation and test sets to ensure that they still had the real-world conditions.

3.6 Hyperparameters

For newResNet18, hyperparameters to be tuned in this step were learning rate, weight decay, dropout rate and batch size.

The learning rate *lr* controlled the size of the steps that the Adam optimizer took during each iteration to update weights. It determined the speed the model would learn from the gradients of the loss function, and it influenced the speed for the newResNet18 model to learn the relationship between structures from OCT slices and the retinal sensitivity.

The weight decay in the context of the Adam optimizer here used the weight update formula [31]

$$w_{t+1} = w_t - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} - \eta \cdot \lambda \cdot w_t$$

where η was the learning rate *lr*, $\hat{m}_t$ and $\hat{v}_t$ were the bias-corrected first and second moment estimates for Adam optimizer, ε is a small constant for numerical stability and λ is the weight decay parameter.

The dropout rate p was the probability of dropping out a neuron used the formula [17]

$$h_i' = \frac{h_i}{1-p} \text{ with probability of } 1-p$$

where h_i is the output of the neuron before dropout and h_i' is the output of the neuron after applying dropout.

Additionally, the batch size for newResNet18 would affect how efficiently the model processed the OCT slices and updated its weights.

Optuna was used for hyperparameter tuning here. First, the deep learning model was initialized using the parameters suggested by the Optuna trial [32].

The model was trained over several epochs, with the Mean Squared Error (MSE) loss function being used to calculate the difference between predicted and true values:

$$L_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

where y_i represented the true values and $\hat{y}_i$ denoted the predictions. The weights were updated after each mini-batch using backpropagation, with gradients computed and passed through the optimizer. During each epoch, the training loss was accumulated, and messages were printed periodically to monitor the training process. The learning process adjusted the weights such that the error between the model predictions and the true labels minimized over time.

Throughout training, the performance of newResNet18 on a validation set was evaluated after each epoch to monitor potential overfitting. The validation loss was computed using the Mean Absolute Error:

$$L_{\text{MAE}} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

If the validation loss did not improve for several consecutive epochs, early stopping was triggered. This technique ended training when no improvement in the validation set was observed, ensuring that the model did not overfit. The patience for early stopping was set at 5 epochs, meaning that training stopped if the validation loss failed to improve for 5 epochs.

In parallel, Optuna evaluated the performance of each hyperparameter trial based on the MAE validation loss after every epoch. The Median Pruner was used to compare the current trial's validation loss against the median loss of all previously completed trials [32]. If the current trial's performance fell below the median, trial pruning was triggered to stop that trial early. This allowed Optuna to focus

on more promising hyperparameter configurations, saving computational resources and reducing overall training time.

Both early stopping and trial pruning worked together to ensure efficient training. Early stopping prevented overfitting [33] within individual training runs, while trial pruning ensured that poorly performing hyperparameter trials were terminated early to optimize the search process [32].

Optuna used the Tree-structured Parzen Estimator (TPE) as the sampling strategy to guide the exploration of hyperparameters across multiple trials [32]. TPE created two models, one for configurations with losses better than a threshold and another for those worse than the threshold. This allowed the optimizer to sample hyperparameters from areas of the search space that were more likely to reach better results.

The hyperparameter search continued over 20 trials, with the result of every trial printed. The trial that obtained the lowest validation loss was considered the best, and its corresponding hyperparameters were returned as the optimal configuration for the model.

The goal of tuning those hyperparameters using Optuna was to find the optimal configuration that could maximize the performance of newResNet18 model while minimizing the risk of overfitting or poor convergence. After tuning those hyperparameters, the model could then be trained with those hyperparameter values applied. However, the total training epoch was not necessarily 20, it should be determined after analyzing the results of running Optuna.

After deciding the training epoch number for the model newResNet18, a similar-structured Optuna code was run for LeNet5, with maximum epoch number set exactly at the training epoch of newResNet18. Same as for newResNet18, the hyperparameter set resulting in the lowest MAE validation loss was chosen. The only difference between the Optuna code for LeNet5 and newResNet18 was the lack of dropout rate parameter range setting for LeNet5 since there was no dropout layer in the model.

Chapter 4

Results and Evaluation

4.1 Results of Data Preparation

In total, there were 86 pairs of MP SLO image and its corresponding OCT SLO image before sifting. After human classification on whether the matching was successful or not using threshold factors of 1, 1.5 and 2, i.e., whether the blood vessels overlapped in the overlay image combining of the currently registered MP SLO and the OCT SLO, Table 4.1 was generated for results.

Threshold Factor	Successful	Failed	Success Ratio
1	46	40	53.49%
1.5	49	37	56.98%
2	46	40	53.49%

Table 4.1: Results of setting different threshold factors, showing the number of successful and unsuccessful matches plus the ratio of successful matching over the total number of pairs

According to Table 4.1, setting the factor to 1.5 maximized the success ratio among all the threshold factors. Then the slope threshold formula could be updated by

$$[\text{Median Slope} - 1.5 \times \text{IQR},\ \text{Median Slope} - 1.5 \times \text{IQR}]$$

Using such formula, there were 37 failed matches and 49 successful matches. Those 37 failed matches were then registered manually using Adobe Photoshop.

As shown in Table 4.2, the final splitting ratio of B-scan slices was approximately 78 / 10 / 12 for the training / validation / testing datasets, and the total number of OCT slices generated from all the OCT images was 121,551. Although the final splitting ratio was not exactly 80 / 10 / 10, it was close to the expected ratio and could be accepted.

Training	Validation	Testing	Total
95,054	12,154	14,343	121,551
78.20%	10.00%	11.80%	-

Table 4.2: Result of OCT B-scan slices splitting across training, validation, and testing sets with corresponding ratios

4.2 Results of Hyperparameter Choosing

When hyperparameters were tuned using Optuna, 20 trials of each trial containing 20 epochs were run for newResNet18. Among all 20 trials, none of them was completed, and every trial was pruned after 6 to 12 epochs, which indicated that there existed either bad hyperparameter choices or model overfitting. Finally, a set of hyperparameters were chosen as shown in Table 4.3, such hyperparameter set of newResNet18 generated lowest validation MAE loss. The hyperparameters for LeNet5 in Table 4.3 were also selected based on Optuna by running a similar code again with the maximum epoch number setting at 12.

Model	Learning Rate	Batch	Dropout Rate	Weight Decay
newResNet18	7.36×10^{-5}	32	0.29	0.30
LeNet5	8.45×10^{-4}	35	-	0.19

Table 4.3: Hyperparameter values used in newResNet18 and LeNet5 models, numbers were all rounded to 2 decimal places. Hyperparameters for newResNet18 and LeNet5 were both chosen using Optuna.

4.3 Model Evaluation

Since all the trials were pruned after 12 epochs in the process of Optuna hyper-parameter tuning for newResNet18, the training epoch number would be set to a number close to 12 to prevent further overfitting. The training epoch number was eventually set at 12 since 12 training epochs reached the lowest testing MAE loss as shown in Table 4.4 and Figure 4.1. This table and plot illustrated that the peak of newResNet18 model learning was at Epoch 12 since there was a noticeable drop in MAE, and it started to overfit since Epoch 13.

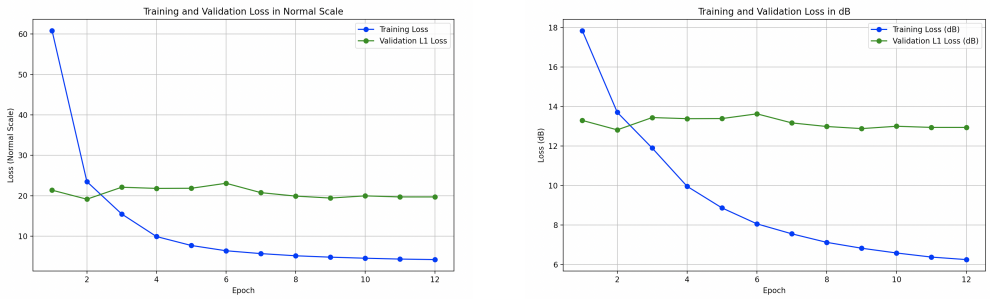
Training Epoch	Testing Loss (MAE)
14	5.48
13	5.40
12	4.50
11	5.47
10	5.29
9	5.40

Table 4.4: Different training epochs and their corresponding testing loss (MAE)

Figure 4.1: The plot showing the testing loss (MAE) across different training epochs from 9 to 14.

The newResNet18 model and LeNet5 model were both trained for 12 epochs

with hyperparameters in Table 4.3, and the training curve for the newResNet18 model was shown in normal scale Figure 4.2 (a) and dB scale Figure 4.2 (b). In those plots, MSE was used for training loss, and MAE was used for validation loss. In the training curve plot Figure 4.2, the training MSE loss descended dramatically in the first epoch and then decreased in a smooth and slow motion in the remaining epochs. The final training loss reached at approximately 6 dB, and the validation MAE fluctuated up and down in a small range between 12.8 dB and 13.7 dB during the whole training process as shown in 4.2 (b).



(a) (b)

Figure 4.2: Learning curves of newResNet18 for training with mean squared error plus mean absolute error L1 loss as the loss function for validation. The losses are in normal scales in (a) and dB scales in (b).

Figure 4.3 shows the scatterplot with real observed MP sensitivity values in x-axis and model-estimated sensitivity values in y-axis, the scatterplot is generated using data from test set. Every point is represented by a hollow circle. The range of model-predicted sensitivity values is from 5 dB to 28 dB approximately, while the range of observed sensitivity values is actually between 0 dB and 33 dB. It is noticeable that for OCT slices with high MP sensitivity values, the model has a higher chance of predicting a larger number, which could be observed from a vertical high-density line at the observed sensitivity of 25 dB for instance. However, the model's ability of predicting small sensitivity values is poor, especially for OCT slices with 0 dB sensitivity value.

Using the formula of Pearson correlation r:

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2 \sum_{i=1}^{n}(Y_i - \bar{Y})^2}}$$

where n is the total number of paired data points in the test set, X_i is the i-th value in the observed sensitivities, Y_i is the i-th value in the predicted sensitivities, $\bar{X}$ demonstrates the mean of the observed MP sensitivities, and $\bar{Y}$ is the mean of the model-predicted sensitivities. After calculation, the Pearson correlation of newResNet18 equals 0.64, showing a moderate to high positive relationship between the predicted sensitivity values and the observed MP values.

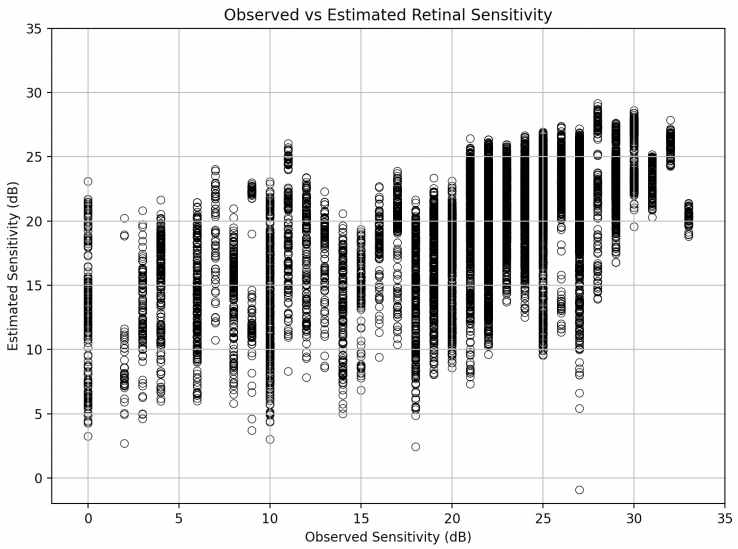


Figure 4.3: The scatterplot of the predicted retinal sensitivity values by the model against the observed retinal sensitivity values obtained from the microperimetry

The Bland-Altman plot was generated as well, as shown in Figure 4.4. This plot shows a small bias since the red horizontal line indicating the mean difference between the observed and estimated sensitivity values is slightly above 0 dB, which suggests that the model-estimated sensitivity values tend to be a bit higher than the real observed values. By observation, the plot also suggests the large variability in the accuracy. In addition, most of the points fall into the 95% CI range bounded by two gray dashed lines, which indicates that the majority of the observed-and-estimated differences are in a reasonable range. However, there are abundant points in the bottom left corner which are not located within the range formed by two gray dashed lines, which means that the model behaves poorly when the average of observed and estimated sensitivity is low, especially when the average sensitivity value is around 10 dB.

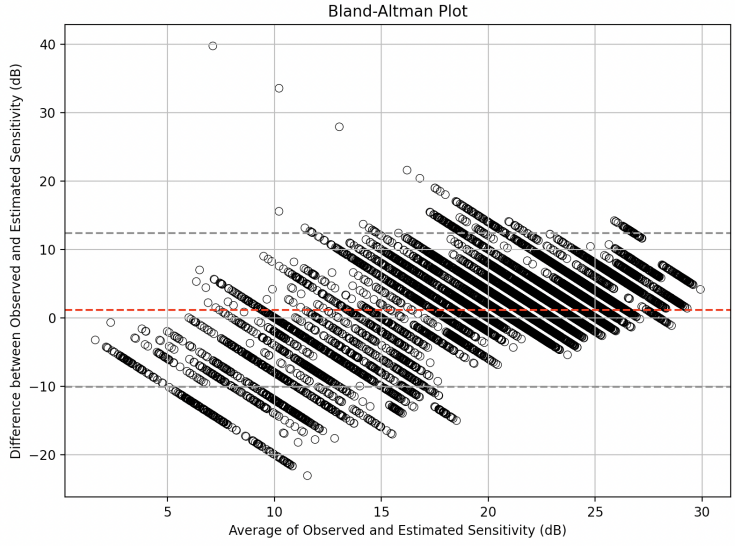


Figure 4.4: Bland-Altman plot showing the difference between real observed MP sensitivity values and model-estimated sensitivity values against their averages. The red dashed line shows the mean difference. The gray dashes draw the range of 95% CI.

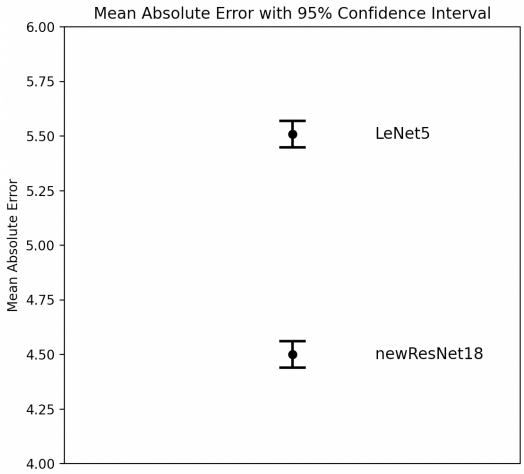


Figure 4.5: Mean absolute errors and its 95% confidence interval for trained models newResNet18 and LeNet5, values are in normal scales.

Next, the mean value error of the newResNet18 model on the test set is calculated using the equation

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

where y_i is the observed true sensitivity value, $\hat{y}_i$ is the sensitivity value predicted by newResNet18, and n is the number of data points in the test set. This formula

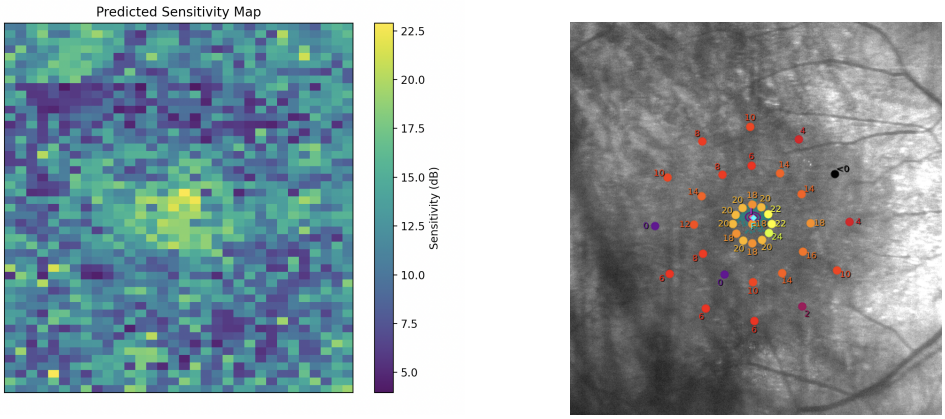
calculates the average of the absolute differences between predicted and true values. Similar to the sensitivity equation, transferring MAE to the dB scale uses the same formula:

$$\text{MAE in dB} = 10 \times log_{10}(\text{MAE})$$

After calculating it in dB, the MAE value for the model newResNet18 is 6.53 dB, with a 95% confidence interval (CI) from 6.47 dB to 6.59 dB. Same information of LeNet5 was calculated as well, the MAE is 7.41 dB with a 95% CI from 7.36 dB to 7.46 dB when using the classic LeNet5 model. All the MAEs with its 95% CIs in the normal scale is shown in Figure 4.5. By paired Wilcoxon rank sum test [34], P is less than 0.001, which also indicates that the newResNet18 model performed significantly better than the baseline model LeNet5.

4.4 Visualization of Prediction

The predicted sensitivity map visually shows the sensitivity values estimated by the model. The generating process of such map was similar to the thickness map Figure 3.2(a). Each 1024×496 OCT volume image was cut into exactly 32 vertical 32×496 thin slices. After passing all the OCT slices as inputs to our ResNet18 model, estimated sensitivity values (in dB) would be obtained.



(a) (b)

Figure 4.6: (a) The sensitivity map generated using predicted sensitivity values by our model. The OCT image inputs used to predict values here are the same as those used in Figure 3.2(a).
(b) A cropped MP image to match positions of (a).

Figure 4.6 (a) was created from the same OCT images used in the thickness map Figure 3.2 (a). By observation, a vaguely visible dark-coloured ring is surrounding a small bright circle in the center, which is similar to the thickness map Figure 3.2 (a).

Comparing 4.6 (a) and its corresponding MP image (b), the accuracy of the sensitivity map is not high. Although the general pattern of the sensitivity map matches the observed sensitivity values, there are some MP points that have unmatched corresponding sensitivity. For instance, the point with observed MP value 24 dB in the inner circle has a predicted sensitivity value only around 12 dB.

Chapter 5

Discussion

5.1 Comparison with Other Function-Prediction Projects

This study used different methods comparing to other similar studies which also focused on predicting retinal sensitivity or visual acuity using deep learning models. In Inoda et al.'s study [14], GoogLeNet was used as the base model in order to handle complex image features. Data augmentation they used included horizontal flipping and random cropping. In Kihara et al.'s study [6], VGG16 was modified and trained. Data augmentation they used included horizontal flipping, rotation and random image translation.

In this study, a different deep learning model ResNet18 was selected for its residual connection and proper depth. The same slice size Kihara et al. used, 32×496, was chosen. However, the slice generating technique was different from their technique. In their study, slices in training data were created using the overlapping technique, but such method was not for validation or testing datasets, which meant that there were abundant similar data in the training set since the overlapping allowed a lot of slices to be chosen for one single microperimetry stimulus point, but few in validation and testing sets. In our study, the overlapping technique was used for all the three datasets, thus there were a lot of similar data in the validation and testing sets as well, which could raise the overfitting issue and hurt the generalization by having too much similar data in the validation set [35], but it would

benefit the model performance since it would make data distribution of the training set and the testing set to be similar [35].

Additionally, the data augmentation technique used in this study was different from any of Inoda et al.'s and Kihara et al.'s. Training data was split into 2 groups based on their sensitivity values, and different strength of data augmentation was applied to those 2 groups with different data frequency. Both of those two data augmentation techniques included horizontal reflection and rotation, but the range of rotation degree was different. The probability of applying those two data augmentation was the same for two data groups, so the data distribution in training set was not distorted. This strategy could force the model to learn low-frequency data.

5.2 Limitation and Probable Improvements

Pearson correlation r of newResNet18 equals 0.64, which shows a moderate to high degree of agreement on the predicted value and observed sensitivity values. Such correlation indicates that the model's performance is good, but there is still room for improvements.

First, the dataset used in this project had a small size, after sifting, there were only 82 pairs of MP SLO and OCT SLO images available to generate OCT slices with perimetry values. The small dataset directly influenced the model's ability to learn based on insufficient training data. Although a series of data augmentation was applied in order to increase the generalizability of the model, the datasize was still not large enough.

In addition, although heavy augmentation was applied to low frequency data and light augmentation was applied to high frequency data, the probability of applying those two different augmentation techniques were both set at 75% in order to keep the distribution of training data unchanged, there could still be a strong lack of OCT slices with sensitivity values smaller than 20 dB. Thus, the predicting ability of the model was assumed to be weak on low sensitivity values.

In machine learning cases on medical imaging, data leakage should be prevented by ensuring there is no data from the same patient existing in different

datasets simultaneously, or the model would learn patient-specific features from training data rather than generalizable structures [28] [29]. Since the datasize was not large enough to support the patient-level data splitting method, it was determined to only do a volume-level data splitting method, which stands for ensuring there was no data from the same OCT volume scan image in different sets. Such method could lead to data leakage since the model was still able to learn patient-specific features.

All those limitations mentioned above could be prevented using a larger dataset, which would assist the model to learn from sufficient different varieties of data, and it could support the patient-level data splitting method and prevent data leakage.

This study only used data based on the single disease Retinitis Pigmentosa, thus the model might not be generalized to all retinal diseases. Additionally, since RP has many varieties including different gene mutations [2], the data was not sufficient for the model to generalize to and perform well on all kinds of RP since the data was collected from patients with USH2A and MYO7A mutations only. It still requires significant data and further development before it can support clinical decisions applying to patients with RP. Using data collected from patients with different kinds of disease could improve its precision.

During the process of image registration, there was manual inspection and matching participating, and those were applied to approximately 43% of MP SLO and OCT SLO pairs. It requires a lot of effort, and it is also time-consuming. Besides, the precision level of manual registration was not high if comparing to machine registration, and any error during this process would result in a mismatch between the microperimetry stimuli point and the OCT B-scan slice, which impacts the correctness and reliability of training data the model receives.

As shown in Figure 4.6(a), there was an obvious mosaic pattern existing in the whole predicted sensitivity map that was generated using the model-estimated sensitivity values, which indicates that the clarity and resolution of such map is not high. If using OCT B-scans of volumes with a larger number, i.e., conducting more

B-scan volume imaging during one OCT test, the map will be composed of a lot more squares compared to the current condition.

5.3 Conclusion

Since retinal structures are strongly related with sensitivity values, a deep learning model was trained using microperimetry and OCT B-scan images in this study. After training, such model is able to estimate retinal sensitivity value based on a slice of OCT B-scan volume image. Based on values predicted by the model, a sensitivity map could be generated on each eye. Compared with microperimetry, the generated map has a larger coverage and higher resolution, and the process of obtaining such map is less time- and effort- consuming. The model could be potentially used as part of a clinical trial for pre- and after- treatment comparison.

Chapter 6

Appendices

The code used in this study is saved in the link:

https://liveuclac-my.sharepoint.com/:f:/g/personal/ucab195_ucl_ac_uk/EqyTDnwyczdCtMeN12qUI0YBOCd3dluaNMaoOD2sYAgbrQ?e=V3bB4A

Bibliography

[1] D. R. Lucas. Retinitis pigmentosa: Pathological findings in two cases. *British Journal of Ophthalmology*, 40(1):14–23, Jan 1956.

[2] J. Kaplan, D. Bonneau, et al. Clinical and genetic heterogeneity in retinitis pigmentosa. *Human Genetics*, 85:635–642, 1990. October 1990.

[3] Maria Toms, Adam M. Dubis, et al. Clinical and preclinical therapeutic outcome metrics for ush2a-related disease. *Human Molecular Genetics*, 29(11):1882–1899, July 2020.

[4] Jasleen K. Jolly, Thomas L. Edwards, et al. A qualitative and quantitative assessment of fundus autofluorescence patterns in patients with choroideremia. *Investigative Ophthalmology & Visual Science*, 57(9):4498–4503, August 2016.

[5] Zsuzsanna Szepessy, Mirella T. S. Barboni, et al. Retinal sensitivity and fixation stability changes during repeated microperimetry. *Journal of Clinical Experimental Ophthalmology*, 8(6), 2017. December 06, 2017.

[6] Yuka Kihara, Tjebo F. C. Heeren, et al. Estimating retinal sensitivity using optical coherence tomography with deep-learning algorithms in macular telangiectasia type 2. *JAMA Network Open*, 2(2):e188029, 2019.

[7] Chelsea E. Myers, Barbara E. K. Klein, et al. Retinal thickness measured by spectral domain optical coherence tomography in eyes without retinal abnormalities: the beaver dam eye study. *American Journal of Ophthalmology*, 159(3):445–456.e1, 2015. Published online 2014 Nov 25.

[8] Jasdeep S. Gill, Vasileios Theofylaktopoulos, et al. Investigating biomarkers for ush2a retinopathy using multimodal retinal imaging. *International Journal of Molecular Sciences*, 23(8):4198, April 2022.

[9] Eva Lenassi, Eric Troeger, et al. Correlation between macular morphology and sensitivity in patients with retinitis pigmentosa and hyperautofluorescent ring. *Investigative Ophthalmology & Visual Science*, January 2012.

[10] Franziska Georgia Rauscher, Panagiotis Azmanis, et al. Optical coherence tomography as a diagnostic tool for retinal pathologies in avian ophthalmology. *Investigative Opthalmology & Visual Science*, November 2013.

[11] Adam M Dubis, Ahmed M Hagag, et al. Structure/function correlations in ush2a- associated usher syndrome. *Investigative Ophthalmology & Visual Science*, June 2022.

[12] Ahmed M Hagag, Andreas Mitsios, et al. Characterisation of microvascular abnormalities using oct angiography in patients with biallelic variants in ush2a and myo7a. *Br J Ophthalmol*, April 2020.

[13] Soung Jun Kim, Chae Hyun Song, et al. Structure-function relationship in patients with retinitis pigmentosa and hyperautofluorescent rings. *Journal of Clinical Medicine*, 2022.

[14] Satoru Inoda, Hidenori Takahashi, et al. An ai model to estimate visual acuity based solely on cross-sectional oct imaging of various diseases. *Graefe's Archive for Clinical and Experimental Ophthalmology*, 2023.

[15] Kaiming He, Xiangyu Zhang, et al. Deep residual learning for image recognition. *arXiv:1512.03385*, 2015.

[16] Geoffrey E. Hinton, Nitish Srivastava, et al. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*, 2012.

[17] Nitish Srivastava, Geoffrey Hinton, et al. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 2014.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, 2012.

[19] Inc. Heidelberg Engineering. Spectralis hra+oct machine, 2006. Heidelberg, Germany.

[20] Adrian Gh. Podoleanu and David A. Jackson. Noise analysis of a combined optical coherence tomograph and a confocal scanning ophthalmoscope. *Applied Optics*, 38(10):2116–2127, April 1999.

[21] CenterVue SpA. Macular integrity assessment (maia) device. Padova, Italy.

[22] Adriana M. Morgan, Livia S. Mazzoli, et al. Expediency of the automated perimetry using the goldmann v stimulus size in visually impaired patients with glaucoma. *Ophthalmology and Therapy*, 8(2):305–311, Jun 2019. Published online 2019 Mar 13.

[23] Maximilian Pfau, Jasleen Kaur Jolly, et al. Fundus-controlled perimetry (microperimetry): Application as outcome measure in clinical trials. *Progress in Retinal and Eye Research*, 82, 2021. Published online 3 October 2020.

[24] D. Odell, A. M. Dubis, et al. Assessing errors inherent in oct-derived macular thickness maps. *Journal of Ophthalmology*, 2011:692574, 2011. Epub 2011 Aug 17. PMID: 21869920; PMCID: PMC3157761.

[25] Marcin Kociołek, Michał Strzelecki, and Rafał Obuchowicz. Does image normalization and intensity resolution impact texture classification? *Computerized Medical Imaging and Graphics*, 84:101716, 2020. Published 6 March 2020.

[26] Xiaolong Pei, Yu hong Zhao, et al. Robustness of machine learning to color, size change, normalization, and image enhancement on micrograph datasets with large sample differences. *Materials Design*, 2023. Published 10 June 2023.

[27] Benjamin Davidson, Angelos Kalitzeos, et al. Fast adaptive optics scanning light ophthalmoscope retinal montaging. *Biomedical Optics Express*, 9(9):4317, 2018. Published Aug 15, 2018.

[28] Gaël Varoquaux and Olivier Colliot. Evaluating machine learning models and their diagnostic value. In Olivier Colliot, editor, *Machine Learning for Brain Disorders*. Springer, 2023. ffhal-03682454v5f.

[29] Elina Thibeau-Sutre, Mauricio Díaz, et al. Clinicadl: An open-source deep learning software for reproducible neuroimaging processing. *Computer Methods and Programs in Biomedicine*, June 2022.

[30] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR)*, 2015.

[31] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[32] Takuya Akiba, Shuji Suzuki, and Keisuke Fukuda. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631, New York, NY, USA, 2019. ACM.

[33] Takuya Akiba, Shuji Suzuki, and Keisuke Fukuda. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631, New York, NY, USA, 2019. ACM.

[34] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[35] Yun Xu and Royston Goodacre. On splitting training and validation set: A comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization performance of supervised learning. *Journal of Analysis and Testing*, 2(3):249–262, 2018.