UNIVERSITY OF CALIFORNIA
RIVERSIDE

Deblurring in Scanning Laser Ophthalmoscopy Using Artificial Neural Networks

A Thesis submitted in partial satisfaction
of the requirements for the degree of

Master of Science

in

Bioengineering

by

Mohssen H. Kassir

March 2022

Thesis Committee:
Dr. Jia Guo, Chairperson
Dr. Hyle Park
Dr. Salman Asif

The Thesis of Mohssen H. Kassir is approved:

Committee Chairperson

University of California, Riverside

ABSTRACT OF THE THESIS

Deblurring in Scanning Laser Ophthalmoscopy Using Artificial Neural Networks

by

Mohssen H. Kassir

Master of Science, Graduate Program in Bioengineering
University of California, Riverside, March 2022
Dr. Jia Guo, Chairperson

Adaptive optics (AO) has enabled in vivo imaging of the living human retina with diffraction-limited spatial resolution and thereby can image the retinal structure at the cellular level. However, AO may not readily be available for all imaging modalities, and in some cases its performance may be compromised by incomplete compensation of ocular wave aberrations. We investigate a deep learning based method to enhance spatial resolution of retinal images without or with AO.

Twelve high-resolution retinal images were obtained using AO scanning laser ophthalmoscopy (AOSLO) as the ground truth. To model the image blur induced by ocular optical defects, we introduced various wave aberrations (with varying Zernike coefficients up to the 6th order). To expand the dataset, the AOSLO images were split into small patches (2820 average patches/image). With 8 different blurring kernels applied, there were 270,736 image patches for training and testing. In application of the

trained networks, the corrected patches were combined to form images of their original sizes. The artificial neural network was based on a U-net architecture. We performed a 4-fold cross validation study with a quarter of images reserved for testing per cross validation.

Normalized mean squared error (NMSE) and structural similarity index measure (SSIM) were calculated for the images before and after correcting for the blurring effects, using the ground truth images as reference. After correction, the NMSE was reduced by 51% on average, from 0.059 ± 0.019 to 0.029 ± 0.012. The SSIM was improved by 155% on average, from 0.22 ± 0.09 to 0.56 ± 0.12. The improvements were significant (by paired t-tests, $p < 0.001$).

In another experiment, we trained the network on non-retinal images and fine-tuned on retinal images using 6-fold cross validation to explore the feasibility of applying transfer learning.

Our results showed that deep learning may be useful in correcting the retinal image blur caused by aberrations. We outline the next possible phase for our work, where we intend to apply deep learning to retinal images taken without AO in order to demonstrate the removal of aberrations through processing in real-world examples.

Table of Contents

List of Figures

List of Tables

Chapter 1: Background and Introduction

1.1 Introduction

Medical imaging is used extensively to gain a visual representation of organs and tissues for medical diagnostic or scientific purposes [1]. One example is scanning laser ophthalmoscopy (SLO), which is used in conjunction with adaptive optics as a non-invasive imaging modality for the living eye [2]. The images produced show the distribution of retinal cells, giving physicians the ability to analyze for potential diseases or treatments [3]. Machine learning algorithms, such as convolutional neural networks (CNN), can be used to modify and process images [4]. The goal of my thesis is to explore the feasibility of augmenting and processing retinal images with machine learning to improve the quality of retinal SLO images and potentially increase its usefulness in various clinical applications.

1.2 Adaptive Optics Scanning Laser Ophthalmoscopy (AOSLO) and its Challenges

SLO enables us to use a scanning laser source and a confocal pinhole to obtain high contrast retinal images [5]. It operates by deflecting a laser beam in 2 dimensions, hence scanning the retina in a raster pattern [6]. The retina is composed of a multi-layered and weakly scattered tissue, featuring photoreceptor outer segments, photoreceptor inner segments, and other layers [2]. Such a tool allows us to visualize individual rods and cones in a human retina, but the problem is that aberrations limit the possible axial resolution [2]. SLO is subject to unclear and imperfect imaging due to optical aberrations [7]. These are caused by irregularities in the corneal surfaces and lens surfaces of the

human eye, which limit the SLO's ability to achieve high resolutions [7]. Adaptive optics, originally used in astronomy, has been found to be useful for correcting aberrations in imaging systems [8]. It was integrated into the SLO by Roorda et al. to create an AOSLO that can correct the aberrations of all individuals [5]. With ocular aberrations corrected, it becomes possible to achieve diffraction-limited images using AOSLO, increasing the type of applications scanning laser ophthalmoscopy can be used for [7].

AOSLO images give high resolution; enough to visualize rods and cones. Images containing photoreceptors of a retina are used to evaluate density, average size, and reflectivity using intracellular distance, cell density, and other metrics [3]. These can provide diagnoses and treatments to patients [3]. AOSLO is additionally used to monitor retinal changes and temporal processes, such as the relation between retinal disease progressions and degenerations [9].

1.3 Potential Applications from Machine Learning

Machine learning is an effective tool for computer vision problems when large amounts of data is available [10]. As a versatile and powerful neural network structure, CNNs have been used for detection of phenomena and automation of diagnoses within medical (including retinal) imaging [10]. For example, CNNs have been used to create an automated segmentation of microaneurysms within the AOSLO image [11].

Specific neural network architectures already developed can be used for new machine learning problems [10]. The U-net architecture is effective at biomedical image segmentation [12] and the generative adversarial network (GAN) is useful at creating

high quality images based on low quality images [13]. Such architectures could possibly be utilized in our deep learning project.

While some researchers have focused on detection and classification [8], we anticipate a possibility in applying deep learning algorithms to improve the quality of AOSLO images. This may assist us in widening access and confronting challenges from adaptive optics. It may allow the production of high quality images with minimal aberrations with or without implementation of adaptive optics in SLO.

1.4 Potential Issues Alleviated with the Use of Machine Learning

Usage of adaptive optics in clinical applications (and outside research laboratories) has been slow [14]. Challenges prevent a wide adoption of AOSLO. Low fidelity and noise may impede the AOSLO's wavefront sensor, which measures the eye's aberrations, leading to improper corrections. This prevents complete compensation of the pupil's wavefront error [14]. In addition, the imaging system can have a limited field of view [15]. Higher costs and complexities for integrating adaptive optics into instruments also contributes to low accessibility [14].

AOSLOs have become more technologically sophisticated with time [5, 16]. Updates can improve the ability of AOSLO to compensate for wavefront aberrations [14]. However, in addition to requiring labs to acquire new AOSLOs to benefit from the updates, they can be cumbersome to maintain [11]. Integrating machine learning into the AOSLO can help solve this and widen access to high quality AOSLO images globally (at little or no additional costs or resources).

1.5 Major Goal of the Project

The main goal of my thesis is to design a new machine learning based processing pipeline for improving their quality of retinal images. Our goal is to transform SLO images and low quality AOSLO images into higher quality images without the need for a new AOSLO instrument. This can make high quality images far more available globally to physicians and researchers, increasing their utilities and reducing the cost substantially.

Chapter 2: Experimental Design and Pipeline

2.1 Introduction

This chapter describes the pipeline we can use to improve image quality through machine learning. At this stage, before applying to real world retinal images with aberrations, we seek to create a proof of concept. We design a deep learning algorithm that can increase the resolution of artificially blurred AOSLO images. Such an algorithm could be used independently from the ophthalmoscope to improve the image quality using machine learning. While its design will be detailed and explained in this chapter, the experimental results are presented in the next chapter.

The pipeline takes images of varying quality (shown in **Figure 2.1**). These images are blurred to different extents, allowing each original image to have different blurred counterparts. The pipeline then splits the original and blurry images into thousands of small corresponding patches. The blurry patches are fed into the trained CNN, which individually processes them to remove the blurry effect. Next, the processed patches are stitched back together, forming complete, unblurred retinal images closely resembling the ground truth versions. We use high quality images taken at detailed resolutions as the ground truth. We experiment with using different cost functions and different training quantities for our neural network. To measure our success, we evaluate our results using common image quality metrics, specifically mean squared error (MSE) and structural similarity index measure (SSIM).

The trained model provides an effective deblurring tool that can serve as a baseline to compare more sophisticated models developed in the future. This model should be tested on real world examples in a later phase. Such a technology, if provided as a software for physicians and laboratories that only have access to lower-end ophthalmoscopes, may be able to increase the usefulness of those retinal images and the effectiveness of those laboratories to evaluate patients or perform research.

2.2 Methods

2.2.1 Overview of pipeline

The pipeline (**Figure 2.1**) is designed for training and validation and contains all the necessary parts of our software. We start with AOSLO retinal images. For our training pipeline, we used a 4-fold cross validation, where we used 75% of the images for training, and 25% for validation in each fold. Each image is used 3 times for training (for 3 cross validations) and once for testing (for 1 cross validation).

Next, we create blurry images artificially. Artificial convolutional kernels are applied to high quality images. This produces matching pairs of images: a high quality ground truth image, and a blurred, low quality image for training. We can create a set of kernels to apply to training images and another set of kernels to apply to the validation images. Dataset size depends on how many copies of each image we want (each copy with a different blur).

Each training image is split into many small patches (each patch is 32 x 32 pixels) [17]. These patches are inputted into the neural network. The splitting process is applied to both the blurred and ground truth images to have pairs for training. The machine

learning algorithm is based on a CNN. The CNN takes in blurry patches and then outputs processed patches of the same dimensions to calculate loss and backpropagate. Once training is complete, we have a fully trained CNN and can move on to evaluation.

To evaluate our trained network, we modify the pipeline slightly (**Figure 2.1**). We process the blurry patches of our evaluation images through the trained CNN, then stitch them back to form fully processed images. These images should have the blurring effect largely removed. Each processed evaluation image can then be compared to the ground truth image with our two chosen metrics (MSE and SSIM). In the next few sections, we will be covering specific aspects of each step of the pipeline and the reasoning behind our design.

All components of the pipeline were written in Python 3.7, and the machine learning script was written in Tensorflow 2.0.

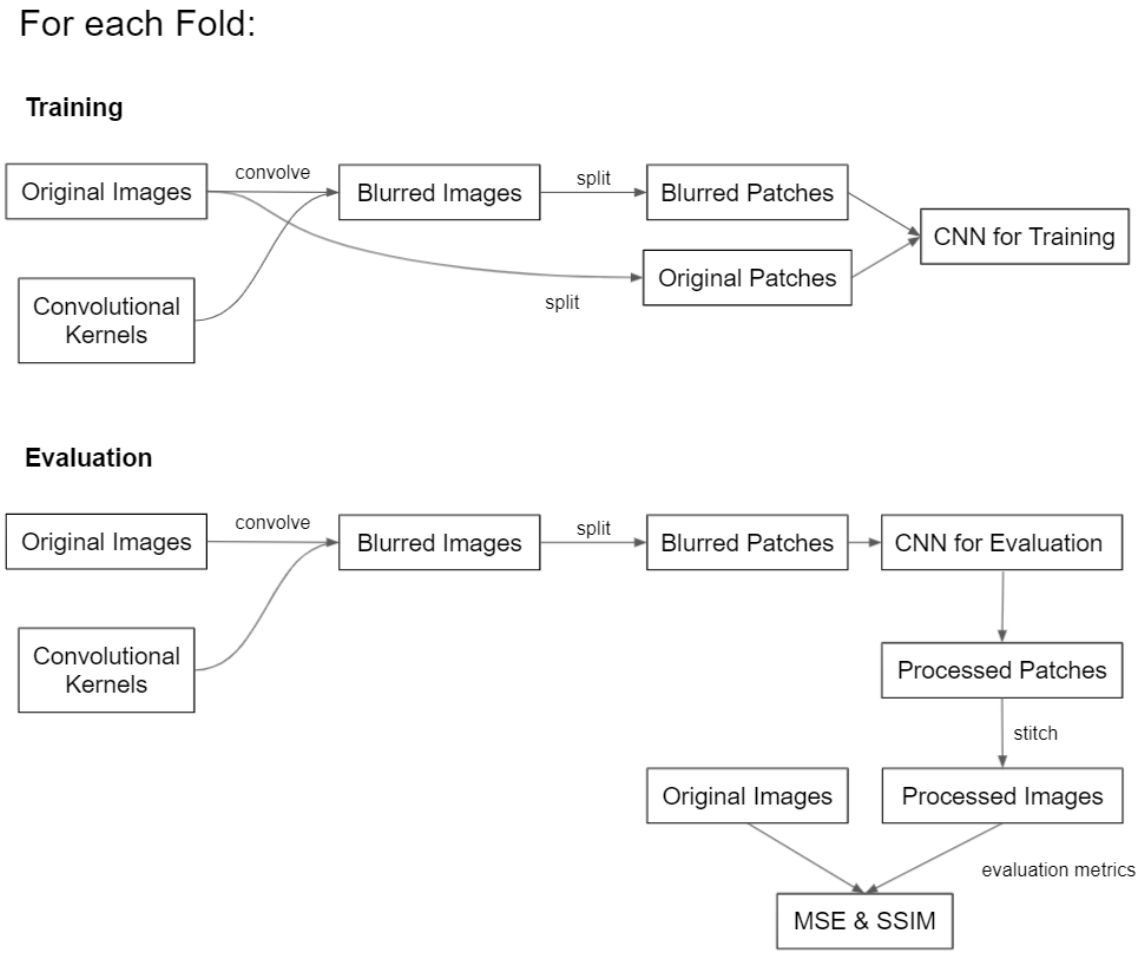

Figure 2.1: Our complete pipeline used a 4-fold cross validation. The training and evaluating portions are shown separately.

2.2.2 Data collection and description

Our dataset consists of 12 high resolution retinal images. 3 of them were provided to us by Dr. Yohua Zhang at the UCLA Doheny Eye Institute. They were acquired with an AOSLO developed with their developed AO-NCO [18]. 9 others were taken from various papers about AOSLO that provided images with their studies [18-21]. These images were of varying sizes, from 688 x 407 pixels to 460 x 254 pixels. Each image was a subsection of a retina, some focusing on areas of high rod content and others focusing

on high cone content. This ensures our ability to work with both rods and cones, as they are the typical targets in AOSLOs [22]. Each of the 4 folds consisted of 9 images for training and 3 for testing. They were randomly split between the 4 folds. 36 images were used in total across the 4 folds for training (3 times per image). 12 images were used for testing, meaning each image of the set was used once across all validations.

2.2.3 Artificial Blurring

The wavefront error seen in many SLO imaging systems is a result of optical aberrations caused by the eye lens' imperfections [2]. We applied artificial blurs to our AOSLO images through convolution to replicate the effect of wavefront error on retinal images. These blurs were modelled after the aberrations that adaptive optics is meant to correct by using Zernike polynomials [2, 23]. Zernike polynomials are a complete set of orthonormal functions over a unit circle. A series of independent Zernike orders can be used to model a complex continuous surface [2]. The first few orders are especially useful because they directly correspond to common optical aberrations such as defocus and astigmatism [2]. We used Zernike polynomials to create a wavefront error function, took the point spread function (PSF), and convolved the PSF with each image [23].

To obtain the proper PSF to convolve with our image, we used the wavefront error function w, which is a weighted summation w of Zernike polynomials.

$$w(x', y') = \sum_{n,m} c_n^m z_n^m(x', y')$$

Each coefficient c was drawn from a Gaussian distribution with a mean matching that of an example pupil with a 3.00 mm radius [24] and an arbitrarily chosen standard

deviation of 0.4. That pupil radius was used to keep computation time reasonable while being realistic. We used Zernike polynomials only up to the 6th order because higher orders are shown to have negligible effects [24]. This wavefront error function and designated pupil area p were used to find a generalized pupil function g [23].

$$g(x', y') = p(x', y')\, exp[\frac{i2\pi}{\lambda} w(x', y')]$$

The wavelength λ is assumed to be 500 nanometers, as suggested by Dr. Zhang. Taking the squared modulus of the generalized pupil function produced the PSF h [23].

$$h(x, y) = ||F[g(x', y')]||^2$$

The PSF was scaled by the resolution of the image, which was assumed to be 714 nanometers for all images based on images from Dr. Zhang's lab. We cropped the 21 x 21 pixel center of the PSF for convolution (where most of the PSF's weight is) to speed up the convolving step [23]. The PSF was cropped because the areas outside the center negligibly contributed to the convolution and doing so saves computation time.

$$r(x, y) = h(x, y) * s(x, y)$$

Artificial blurring allows having multiple unique blurs for each image and increases the data size (**Figure 2.2**). The variation in the blurs produced (based on randomly varying coefficients of the Zernike terms) allows for the different blurred images to represent different amounts of aberrations that need to be corrected in images. Each new blur adds 8 new images for training (one per training image). As a parameter, data size can be altered to improve the generalizability of the method.

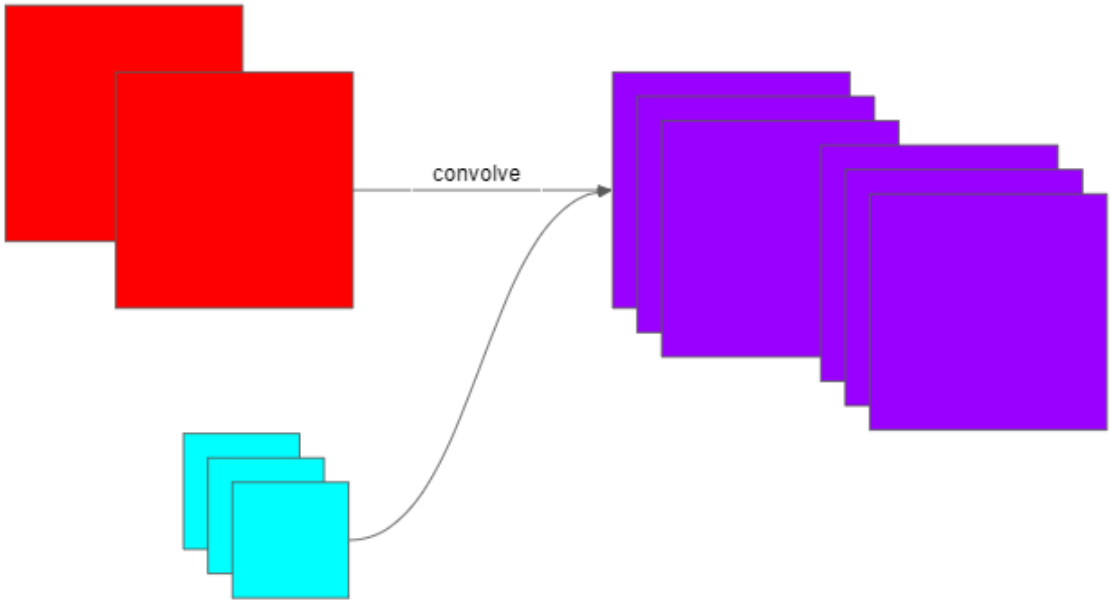

Figure 2.2: The total number of images for training depends on the number of blurry kernels. In this diagram, 3 blurry kernels allows us to multiply the original dataset of 2 for 6 total images.

Our artificial blurring doesn't account for the differences in lighting contrast produced by using AOSLO systems [18]. We chose to not include it in our modelling and instead focus on improving quality through deblurring. This is to ensure we are specifically dealing with and attempting to solve the aberration correction problem, while enhancing lighting contrast is a much easier problem and various solutions are readily available.

2.2.4 Architecture of Neural Network

Multiple architectures were considered and attempted for this project. Focus was mainly devoted to the U-net, Res-net, and GAN architectures. Eventually, we settled on using a U-net with elements of the Res-net (residual connections).

The first attempted architecture was a GAN. Generally, GANs are composed of 2 competing networks, a generator and a discriminator [25]. The generator takes in input

data and seeks to transform it into images that authentically look like the desired images. The discriminator evaluates the generator's produced images and assesses their authenticity. The goal of the generator is to produce authentic-looking images to fool the discriminator, and the goal of the discriminator is to maintain the ability to distinguish. Constant training can produce authentic-looking images [25]. The input in our case is a blurred version of the authentic images, making the deblurGAN possibly useful. The deblurGAN developed by Kupyn et al. can fix motion blurred images [13]. Its generator removes the blur from blurry images [13].

The deblurGAN's advantage is its specialty in dealing with blurs and sharpening images, which was tempting. However, we did not use it because it did not provide successful results in our initial attempts. Controlling two different neural networks and how they interact with each other produced great instability. We instead decided to focus and build a more straightforward and robust neural network with another well-established architecture.

The U-net architecture was primarily developed for biomedical image segmentation [12]. Our network should be able to properly outline all the cellular structures (like segmentation) in order to recover the unblurred form of the images. The U-net, as shown in **Figure 2.3**, first has layers that convolve and shrink the image through max pooling, then convolves and enlarges the image through transpose convolutions. After each upsample when enlarging, results from a step in the shrinking phase are concatenated with the image and are fed into the next function. The resulting output is generally the same or a similar size to the input image [12].

The Res-net architecture was developed to improve the performance of deep neural networks [26]. He et al. found them to be well performing, efficient, and easy to optimize. We used residual connections in our architecture instead of the specific architecture they provide. A residual connection block works by connecting the output of a layer with the input of a layer before it, allowing for residual learning to take place [26].

The full architecture is shown in **Figure 2.3**. It is a 21-layer U-net architecture that takes in 32 x 32 pixel images, performs 2 convolutions and a downsampling step 3 times (until they are 4 x 4 pixel images). To upsample, it then performs 2 convolutions and a transpose convolution. It repeats this upsampling mechanism 3 times. While downsampling, the number of channels reaches 64, and upsampling reduces the channels to 16. Finally, it performs 3 convolutions at the end, and it results with 32 x 32 pixel images.

Each convolutional kernel used in our architecture has a 3 x 3 pixel size [12]. Zero-padding is used to ensure no convolution changes the size of the image. Each max pooling layer reduces the image size by half in each dimension by using a pool size of 2 x 2 pixels and 2 strides. Each transpose convolution uses 2 strides and a 3 x 3 convolutional kernel. Image dimensions are only changed when they are doubled or halved at the sampling steps. There is a ReLU activation function in every layer, except the final layer, which has a tanh activation function [12]. There are 2 total residual connections; both are done before the first 2 upsamplings. We use an adam optimizer with a learning rate of 0.002 [27]. We feed in the input data as batches, each with the size of 1000 patches. During the training phase, we save the weights of the neural network that produces the

lowest cost. Therefore, The final trained network we save uses the best-performing weights and biases.

Images are reduced to 4 x 4 pixel size to maximize our ability to downsample and increase U-net's depth and optimize our training. Careful symmetry in image size is maintained between downsampling and upsampling to allow for transfer of data from the downsampling side to the upsampling side and to maintain dimensions between input and output. The conservation of dimension allows for proper restitching and evaluation.

The current architecture was chosen after a series of experimentations with numbers of layers, channels, and residual connections. A plateau in performance occurred where (for example) additional layers were added and the U-net decreased the size of the images to 2 x 2 pixels. This led to little difference in the performance, but it significantly decreased the speed of the algorithm. It was therefore not seen as optimal.

The small size of input patches in our designed architecture differs from those in traditional U-net applications that use larger images that can be downsampled more [12]. This was chosen based on the observation of similar and repeated structural patterns of retinal cells and the need for larger training datasets due to limited datasets available (see details below).

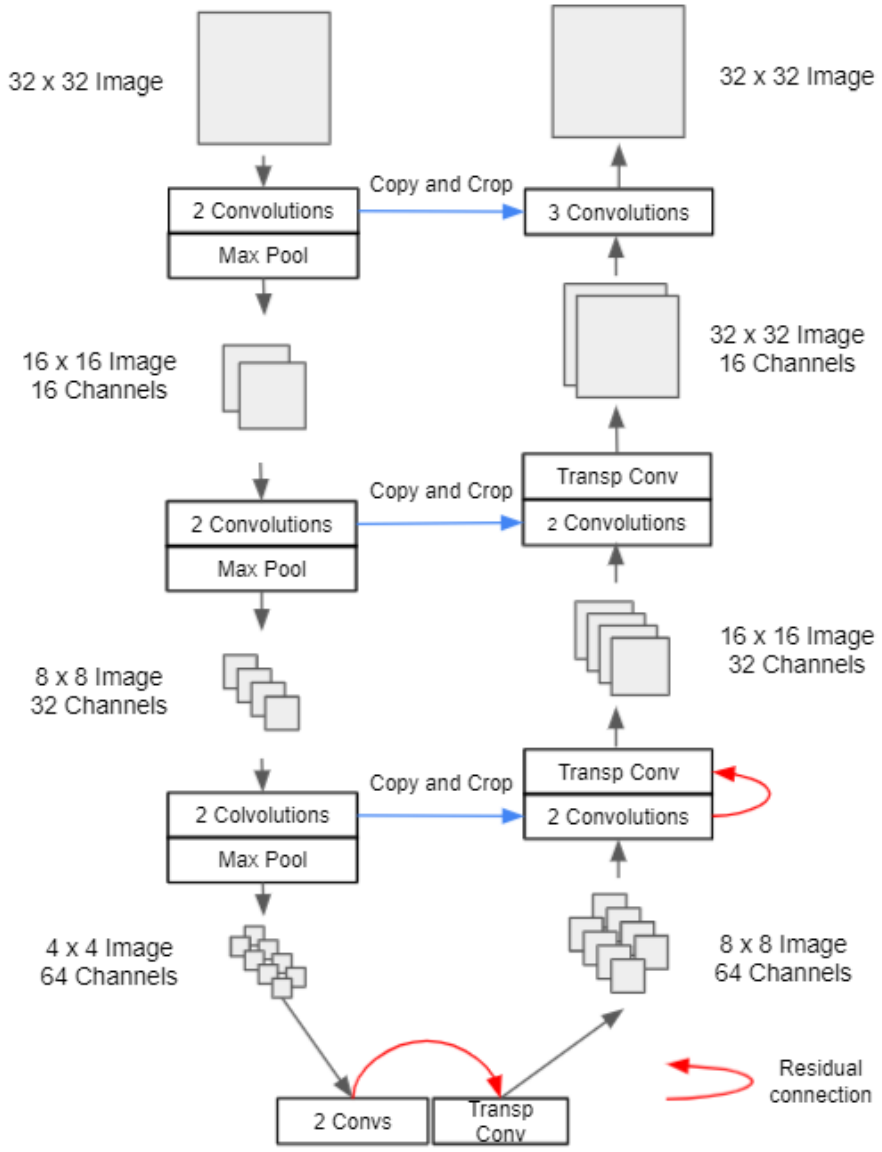

Figure 2.3: The U-net architecture of our neural network

2.2.5 Patching and Stitching

We split the images into many small patches to increase our dataset size and utilize the images' repeated patterns.

Our method of cutting large images down is shown in **Figure 2.4** and explained in the next paragraph. Each image can be used to create thousands of patches, allowing for a

reasonably large dataset useful for the neural network with just 12 images. The cut-down patches still maintain the small and repetitive cellular structures of the photoreceptors, ensuring no loss in structural information.

The algorithm starts in the upper left corner of each image, then shifts 8 pixels to the right to create each new 32 x 32 patch. Once it reaches the right of the top row, it resets to 8 pixels under the upper left corner and continues horizontally cutting patches. The process repeats until the bottom row of the image is taken. If the image dimensions aren't perfectly divisible by 8, the remaining (less than 8) pixels would not be included in any patches, and the original images for evaluation would be slightly cropped accordingly. Every patch is 8 pixels removed from its adjacent patches (vertically and horizontally) and shares 75% of its pixels with each of its adjacent patches. There were approximately 2820 patches per image and an average of 33,842 patches per blurring kernel. Total patches depended on the number of unique blurring kernels. For example, if 8 blurring kernels were used, there would be 270,736 total blurry patches.

Overlapping areas were kept between patches to increase the data size further. While some photoreceptors appear on multiple patches, no 2 patches carry the same pattern due to our method of data augmentation. The number of pixels to shift by was determined after experimentation and comparison of 8, 16 and 32 pixel shifts.

To combine the processed patches, they were stitched back into a whole image. For the overlapping areas between adjacent patches, we needed a way to average all the patch pixels meant to occupy one pixel of the complete image. We tried taking the mean, using the triangular window function, and using the gaussian window function. They all

produced similar results with no significant difference, and the gaussian window function w was arbitrarily chosen.

$$w(n) = e^{-\frac{1}{2}\left(\frac{n}{\sigma}\right)^2}$$

n is the number of pixels we wanted to average, and σ is the arbitrarily chosen standard deviation of 3.

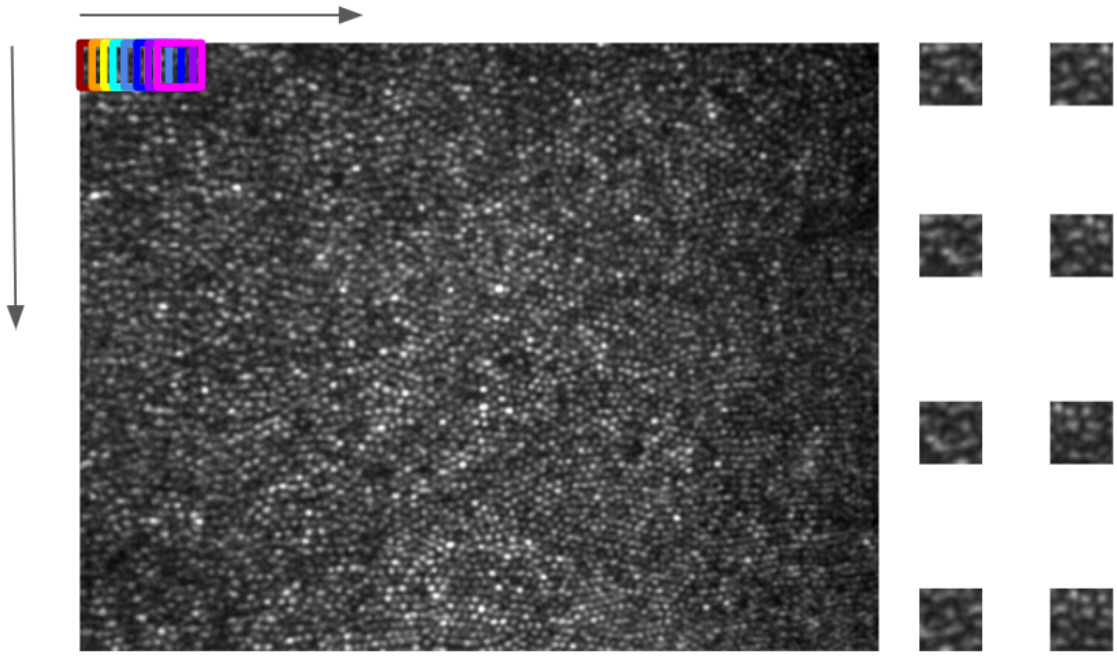

Figure 2.4: Dividing the images into small patches. For each image, we start in the upper left corner and create 32 x 32 patches horizontally, with each patch moved slightly to the right by 8 pixels. After each horizontal row is completed, we move down 8 pixels vertically to continue making patches. This is done until we reach the bottom row of the image. Example patches are on the right.

2.2.6 Metrics for evaluation

SSIM and MSE were used as the metrics to evaluate image quality [28]. This SSIM metric is meant to more accurately reflect the human visual perception system, where quality assessment is based on structural information instead of direct

comparisons. It depends on luminance, contrast, and structural comparisons [28]. It can be considered complementary to the more traditional methods. MSE is another commonly used metric to measure the average pixel differences between images [29]. It evaluates the average squared difference between the estimated and actual value of a sample [30]. The MSE values were normalized (normalized MSE, NMSE) to help training and allow comparison between processings on images of different intensity ranges. While training, SSIM and NMSE were measured over the small patches and combined to be used as a cost function. They were applied over every evaluation image to assess the processed (completely re-stitched) images compared to the ground truth images.

2.3 Advantages and Limitations

Our model gives us a wide variety of blur types to train our neural network on. This makes it generalizable, working across different images taken with different AOSLOs that produce images with varying degrees of quality. As a result, the amount of data we have depends on our parameters, and we can experiment with blur variations. The cross validation allows us to run the neural network multiple times, obtaining an average performance of our pipeline instead of one result to assess its generalisability. All our images can then be used in both parts of our pipeline, providing a more efficient use of the data.

One limitation is the small image set size, and we chose to accommodate that in the data augmentation, hence the usage of overlapping image patches and multiple blurring effects to adjust our dataset size. Our neural networks would be dealing with

many similar input patches because of these 2 factors. Our evaluation section helps address this by testing the effectiveness of the trained model on images (and patches) it has never seen.

2.4 Conclusion

The complete pipeline (with both training and evaluation sections) enables us to test the feasibility of our idea: the use of machine learning to improve image quality due to blurring. Artificial blurring gives us the data needed to work out the concept. Using this with artificial data will provide us with the preliminary results needed to cement the workable structure. Further studies meant to improve the performance would be easier because it would only require a tweaking of the pipeline.

Chapter 3: Execution and Results

3.1 Introduction

The goal of this section is to use the tools developed in the last section to run experiments. We ran the basic experiment to evaluate the pipeline's performance and tried to find the best parameters, such as the cost function ratio. In addition to the evaluation metrics, we also visualized the images for inspection. A successful result allows future improvements by finding ways to improve the network structure or parameters based on testing on different data.

3.2 Methods

3.2.1 Using different amounts of blurs per image

We ran a 4-fold cross validation on 12 images. To test if performance improves with the number of blurry kernels, we ran the experiment multiple times, using different amounts of blurry kernels each time. We ran it using 8 blurry kernels, 16 kernels, and 24 kernels per image for comparison. For each cross validation, with patching, the total size of the training and evaluation set was 270,736 for 8 kernels, 541,472 for 16 kernels, and 812,208 for 24 kernels.

3.2.2 Different ratios for cost function

We ran using the pipeline using different ratios of NMSE and SSIM for the mixed use cost function of the neural network:

$$a * MSE + b * SSIM = \frac{a}{n}\sum_{i=1}^{n}(x_i - y_i)^2 + b * \frac{(2\mu_x\mu_y + 0.0001L^2)(2\sigma_{xy} + 0.0009L^2)}{(\mu_x^2 + \mu_y^2 + 0.0001L^2)(\sigma_x^2 + \sigma_y^2 + 0.0009L^2)}$$

Here, a and b represent the MSE to SSIM ratio, x and y represent normalized arrays of the two image types, μ_x and σ_x represent the mean and standard deviation of x, μ_y and σ_y represent the mean and standard deviation of y, n represents the total number of pixels, 0.0001 and 0.0009 are often used constants in SSIM, and L represents the dynamic range of pixel values [29, 31].

This helped us find the optimal ratio for the best performance. We ran the pipeline with 8 kernels per image starting with 0 to 1.0 NMSE to SSIM ratio. We incrementally increased the value of NMSE by 0.1 and decreased the value of SSIM by 0.1 with each iteration. Our final test was using the ratio of 1.0 to 0 NMSE to SSIM. This gave us a total of 11 experiments to run, all with 4-fold cross validations.

3.3 Results

3.3.1 Using different amounts of blurs per image

Out of the 3 unique 4-fold cross validation experiments, the 8-blur-kernel experiment performed the best (**Figure 3.1**), with an 155% average improvement in SSIM from 0.22 $\pm$ 0.09 to 0.56 $\pm$ 0.12 (paired t-test, $p < 0.001$) on average. NMSE was significantly reduced by 51% on average, from 0.059 $\pm$ 0.019 to 0.029 $\pm$ 0.012 (paired t-test, $p < 0.001$). As seen in **Table 3.1**, the 16-blur and 24-blur experiments performed slightly worse, with 145% average improvements in SSIM, and 42% and 49% average reductions in NMSE, respectively. The cost function had a ratio of 0.8 to 0.2 NMSE to SSIM for all experiments. Paired t-tests were used for statistical comparisons.

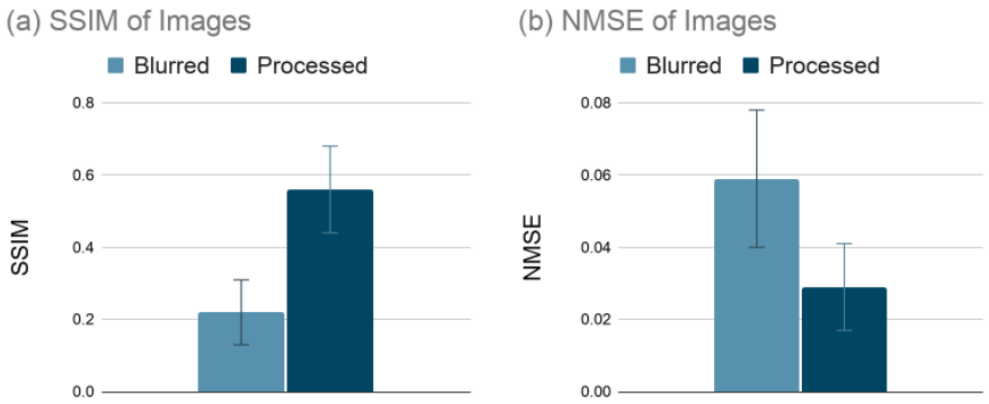

Figure 3.1: Results of a 4-fold cross validation using 8 blurs and a mixed cost function ratio of 0.8 NMSE:0.2 SSIM. (a) The SSIM of the retinal images compared to the ground truth before and after being processed. (b) The NMSE of the retinal images compared to the ground truth before and after being processed.

	Blurred		Processed	
Blurry kernels number	NMSE	SSIM	NMSE	SSIM
8 blurs	0.059 ± 0.019	0.22 ± 0.09	0.029 ± 0.012	0.56 ± 0.12
16 blurs	0.062 ± 0.022	0.22 ± 0.09	0.036 ± 0.022	0.54 ± 0.12
24 blurs	0.061 ± 0.019	0.22 ± 0.09	0.031 ± 0.010	0.54 ± 0.12

Table 3.1: The average NMSE and SSIM of images when using different amounts of blurs before training and after training.

3.3.2 Different ratios for cost function

We tested different ratios of the cost function. We tried using only NMSE, only SSIM, and 9 different ratios using both (**Figure 3.2**). We found 0.8 to 0.2 NMSE to SSIM to be the most optimal, but only by a small margin because most of the results were fairly similar. The ratios of 0.2 to 0.8, 0.6 to 0.4, 0.7 to 0.3, and 0.8 to 0.2 were significantly different from both the ratio of 0 to 1 and 1 to 0 for the SSIM measurements (paired

t-test, $p < 0.05$). None of the MSE measurements were significantly different (paired t-test, $p < 0.05$). Even though the ratio of 0.8:0.2 was the best, other mixed cost function ratios seem to all push the U-net in the same direction compared to the SSIM cost function or the NMSE cost function.

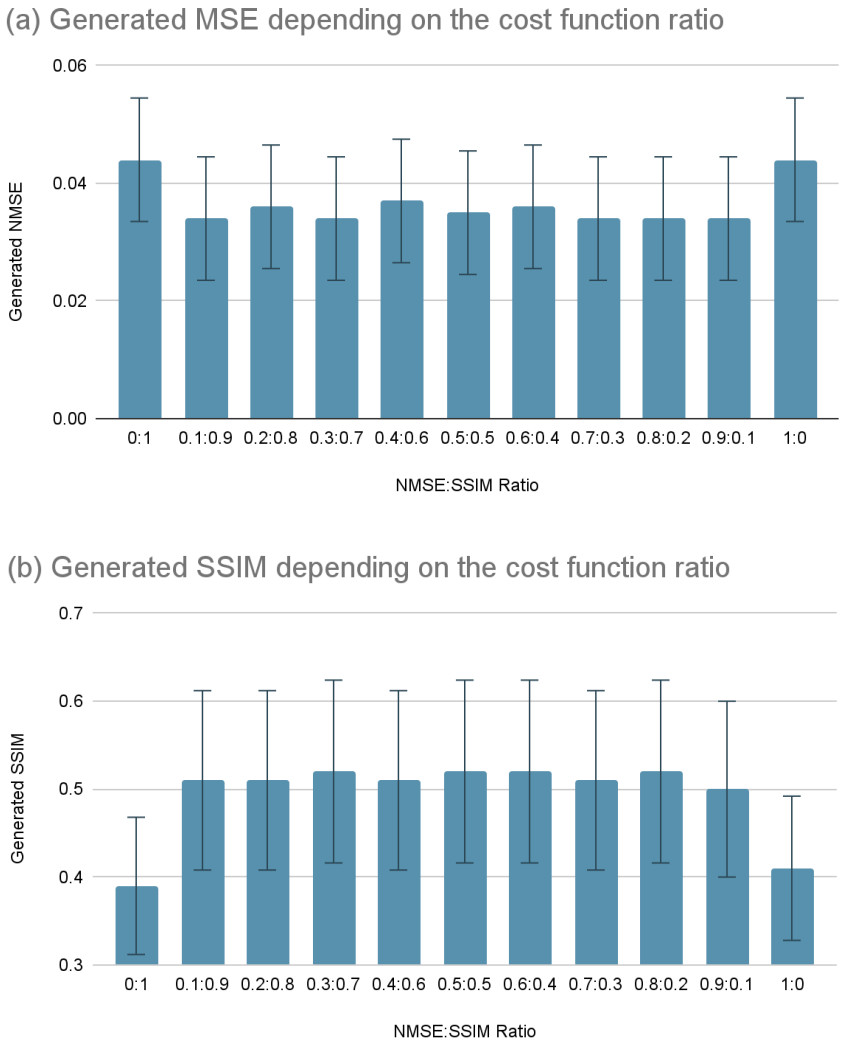

Figure 3.2: The (a) NMSE and (b) SSIM results using 11 cost functions with different NMSE to SSIM ratios.

3.3.3 Example improvements

Both these metrics showed significant improvements in all experiments. **Figure 3.3** and **Figure 3.4** are two example sets of the image improvement as a result of our algorithm. SSIM and NMSE were improved and reduced by 107% and 52%, respectively in **Figure 3.3**. They were improved and reduced by 183% and 47%, respectively in **Figure 3.4**. Visually, the images look much clearer.

Figure 3.3: Examples of: (a) ground truth image; (b) blurry image; (c) corrected image. The NMSE were: (b) 0.042 (c) 0.020, and the SSIM: (b) 0.28 (c) 0.58.

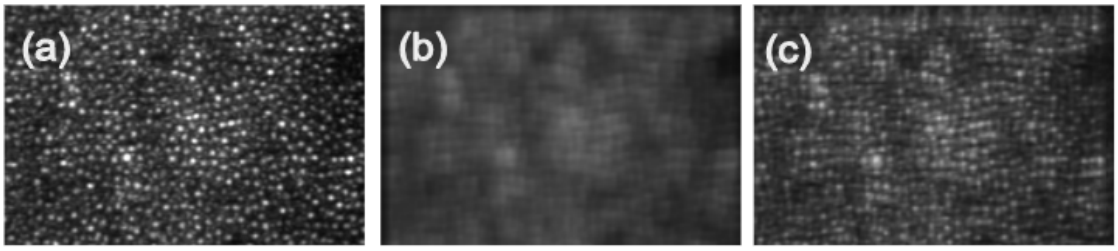

Figure 3.4: Another set of examples. The NMSE were: (b) 0.066 (c) 0.035, and the SSIM: (b) 0.18 (c) 0.51.

3.4 Discussion

We saw a slight decrease in the performance of the neural network when 16 or 24 blurs are used. This was likely showing a plateau in performance. The dramatic increase in time discouraged us from trying to find an optimal blur amount above 8 because the result would likely be only a slight improvement. The cost function used during section

3.3.1 was 0.8 NMSE to 0.2 SSIM. For cost, as seen in section 3.3.2, the mixed cost function seemed to be similarly effective across multiple ratios. However, it was clearly better for it to be mixed than for it to use one metric. Our worst performing cost functions were the non-mixed ones, clearly showing they are disadvantaged.

Visually, we can see in **Figure 3.3** and **Figure 3.4** that the ability to see the individual photoreceptors is generally removed when the blurry kernels are applied. With the processed images, the blur is resolved, allowing us to see them, similar to original images. We acknowledge that the images still appear to be lower quality compared to the originals.

These results provide us with valuable information for further analysis. One way we can evaluate the effectiveness of our image processing pipeline is to evaluate the cell density of each of the images and note the accuracy. We can do so by relying on multiple trained observers counting the number of photoreceptors in each image, giving us the necessary numbers to evaluate [32]. Alternatively, we can rely on newly developed machine learning algorithms developed to automate this process and improve counting accuracy [3].

3.5 Conclusion

Preliminary results show that our pipeline and neural network architecture can properly deblur a low quality AOSLO image. Later sections should draw out how to apply this to more useful data, such as real world examples, to further demonstrate our technology's usefulness.

Chapter 4: Transfer Learning

4.1 Introduction

Transfer learning is often used to improve the performance of machine learning algorithms by transferring knowledge across domains [33]. By using a pre-trained network that was trained on a similar domain, such as related images with a similar format, we can use much of the architecture and optimization from one problem and transfer it to another.

It often becomes useful when there is a shortage of data pertaining to the problem of interest [33]. In this case, many of the patterns seen in similar data would be generalizable and useful on the data of interest [33]. Most of the model's training can come from using similar data. The limited data with the same distribution as the test data can then be used to specify the model to be optimized and applicable to the data of interest.

A pre-trained network can be fine-tuned to the specific problem at hand by removing the task-specific top layers [34]. Instead, these top layer's weights can be retrained according to the more relevant data, helping these task-specific layers become more attuned to the task of interest [34]. This would allow for the use of the knowledge gained from a similar domain to our domain.

Our data may share patterns with other image types which we intend to use for transfer learning to assist our training. This can help our network be more generalizable to various SLO images. Currently we are working with a limited amount of data (12 images). Finding different data types may increase the versatility of our model and

increase our dataset to consist of diverse training examples [35]. This may assist the effectiveness of the neural network and address the data limitation.

AOSLO images are limited due to cost and complexity of equipment [14]. Using transfer learning can mitigate this. Using online images is not optimal, but serves as an important alternative. Many of the images online do not state their resolution or can contain artifacts, so to ensure we use high quality images, we had a limited amount of online retinal data used in our experiment. Datasets which consist of repetitive round structures can potentially be a similar enough domain to use. By primarily training our model on a variety of images of round objects, we hope to then fine-tune our model with AOSLO images to produce a more successful and useful neural network.

4.2 Methods

4.2.1 Using non-retinal images

We included a series of images, fully composed of repetitive small objects, usually round. This image style was meant to resemble retinal images of photoreceptors, which are bright dots juxtaposed against a darker background. As a result, we gathered 15 grayscale images (**Figure 4.1**) of many different types of objects, such as many small bugs crammed together, piles of corn or grapes, or floors of rocks. The large images were scaled down such that a 32 x 32 pixel patch would contain many round objects, just like retinal patches.

The training step used all 15 non-retinal images. The pipeline was the same as the training portion of **Figure 2.1**. We used 8 convolution kernels on the training images (**Figure 2.5**), split them into many small patches, using the same mechanisms as in

Figure 2.4 (8 pixels per shift for 32 x 32 pixel patches), and used the same network architecture as in **Figure 2.3**. The images lead to a total of about 153,000 patches. We evaluated the performance of this trained network (before fine-tuning) on all 12 retinal images, setting the results as the baseline to compare to the results of the transfer learning portion.

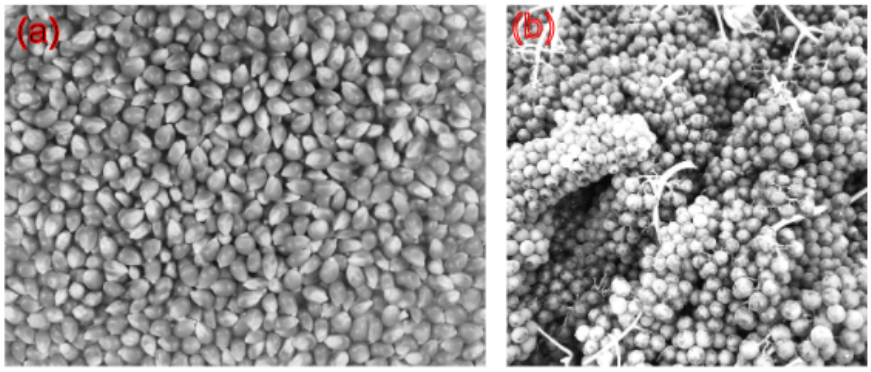

Figure 4.1: Example non-retinal images used for training: (a) corn (b) grapes

4.2.2 Cross validation for fine-tuning

After training and the first evaluation were complete, we fine-tuned the network on the retinal images using a 6-fold cross validation with 8 convolutional kernels applied to each image. Each fine-tuning cross validation was done using 2 randomly assigned images, and we also split them into patches. A 6-fold cross validation allowed for all images to be used while also keeping the data size small (only 2 images) in the fine-tuning step. Each image was used 5 times for evaluation since each cross validation had 10 evaluation images. Each image's metrics were averaged from the 5 images.

4.2.3 Network modifications

We modified the network in 5 different ways and performed 5 different fine-tuning methods (each with a 6-fold cross validation) to assess if fine-tuning could be

used to improve results. Each fine-tuning method modified the neural network architecture in different ways. While the architecture in **Figure 2.3** is mostly kept the same, there are some changes, shown in **Table 4.1**. Trainability, which determines whether the weights of the non-reset layers can still be trained or are frozen during fine-tuning, is turned on in the first 3 methods and turned off in the last 2 methods. If most of the weights are frozen, then only the final layer or two, which usually assess the fine details of an image, are retrained. In the first method, none of the layers had their weights and biases reset to random values. In the second and fourth methods, only the final layer had its weights and biases reset. In the third and fifth methods, the final two layers had their weights and biases reset. We focused on resetting the final layers because they address the fine details of the images (the earlier layers address the general features of the images) [34]. Since the dataset contains images with similar general features, we did not attempt resetting them.

Method	Trainability	Reset Layer Amount
1	on	0
2	on	1
3	on	2
4	off	1
5	off	2

Table 4.1: These different types of experiments were carried out to see the effectiveness of fine-tuning.

Each batch contained 250 patches instead of the usual 1,000. This was done because the fine-tuning set was much smaller, consisting of 2 instead of the usual 9 images. The adam optimizer's learning rate was changed to 0.0005 for this step.

For each experiment, average NMSE and SSIM were found by averaging the results of all 6 cross validations (with 10 images being used for evaluation for each cross validation). Each of the 12 images was used 5 times (in 5 of 6 validations) for evaluation. Because all 12 images are equally used, they can be compared to the results from Chapter 3, where 12 images were used equally (but once each).

4.3 Results

4.3.1 Metrics

The NMSE and SSIM improved during the training phase (**Table 4.2**). The fine-tuning phases in all cases generally showed a decrease in performance. The best performing fine-tuning method is Method 5, which produces the lowest NMSE of 0.0313 $\pm$ 0.0127 (paired t-test, $p < 0.001$) and second highest SSIM of 0.497 $\pm$ 0.118 (paired t-test, $p < 0.001$). This, however, is much lower than the performance before fine-tuning, which is 0.0264 $\pm$ 0.0111 for NMSE (paired t-test, $p < 0.001$) and 0.589 $\pm$ 0.090 for SSIM (paired t-test, $p < 0.001$).

Processing Step	NMSE	SSIM
Blurred	0.0596 ± 0.0198	0.225 ± 0.092
Processed after training	0.0264 ± 0.0111	0.589 ± 0.090
Transfer Learning (TF) Method 1	0.0340 ± 0.0168	0.499 ± 0.134
TF Method 2	0.0342 ± 0.0169	0.496 ± 0.133
TF Method 3	0.0343 ± 0.0169	0.495 ± 0.135
TF Method 4	0.0364 ± 0.0156	0.466 ± 0.121
TF Method 5	0.0313 ± 0.0127	0.497 ± 0.118

Table 4.2: The average NMSE and SSIM of images at different stages and using different methods.

4.3.2 Example Images

Figure 4.2 is an example set of the image change before and during transfer learning. SSIM and NMSE were improved and reduced by 210% and 53%, respectively, in **Figure 4.2c**. They were only improved and reduced by 95% and 34%, respectively, in **Figure 4.2d**. Visually, it appears that the fine-tuned images look slightly worse and that there is more roughness when compared to the images before fine-tuning.

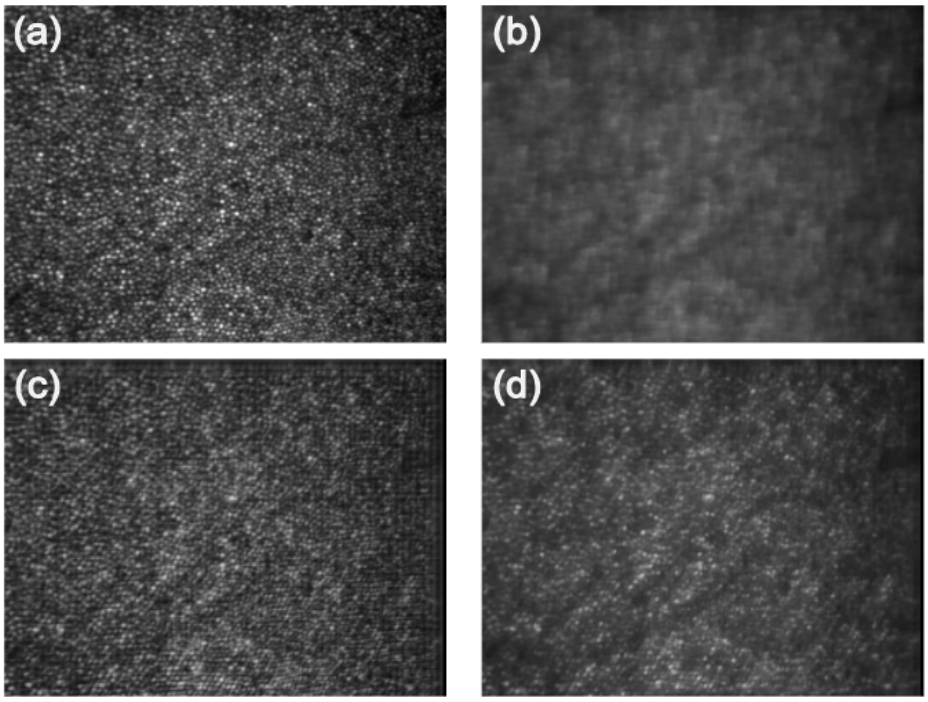

Figure 4.2: Examples of: (a) ground truth image; (b) blurry image; (c) corrected image before fine-tuning; (d) corrected image after fine-tuning. The NMSE were: (b) 0.044 (c) 0.021 (d) 0.029, and the SSIM: (b) 0.19 (c) 0.59 (d) 0.37.

4.3.3 Comparison to results from Chapter 3

The best results are produced from training on non-retinal images. Compared to non-transfer learning with retinal images, transfer learning produced similar NMSE and SSIM. These are models produced under different validation designs.

Experiment	NMSE	SSIM
Trained on retinal images	0.029 ± 0.012	0.56 ± 0.12
Trained on non-retinal images	0.026 ± 0.011	0.59 ± 0.09

Table 4.3: The average NMSE and SSIM of the results from Chapter 3 and the best results from Chapter 4.

4.4 Discussion

The results in **Table 4.2** show that even though transfer learning is effective, the fine-tuning step worsens the performance for evaluation. The image results in **Figure 4.2** is an example showing that the images for both before and after fine-tuning have some level of artifact or blurriness still present. When compared to **Figure 3.3** and **Figure 3.4**, **Figure 4.2d** looks on par, but **Figure 4.2c** looks slightly worse compared to them even though it appears less blurry. **Figure 4.2c** shows a checkerboard artifact, which is not an uncommon problem for U-net and CNNs. It typically results from the upsampling portion of the U-net, which may be the case here [36]. Diagnosing this problem needs further investigation. It may have resulted from the final layers in the U-net, and further tweaking may help resolve it. The cause of the decrease in performance after fine-tuning is also unclear and needs further investigation.

Additionally, non-retinal training is arguably more effective than retinal training, as seen in **Table 4.3** This shows that the diversity brought from incorporating new images into training potentially increases robustness [35]. It may be possible that the repetitiveness of 12 images leads to a level of overfitting, which hurts the network. Another important note is that the sharpness of the images may make them more effective as agents of deblurring compared to retinal images.

4.5 Conclusion

We have explored using transfer learning in this chapter. There has been some success, since there is usefulness for non-retinal images, but it has generally not worked because it leads to worse performance in our chosen metrics. The successful baseline

model from this chapter can be used in future phases in addition to the successful model from the previous chapter. Both models will be used in future phases to test on real-world applications.

Chapter 5: Applying machine learning to real world data

5.1 Introduction

This chapter outlines an experimental design for the future directions. We have shown the effectiveness of deep learning in deblurring artificially blurred AOSLO images but not on real world AOSLO examples. The final step for proving the usefulness of our pipeline is to use real world examples of blurry retinal images. We should compare SLO images taken with adaptive optics being "on" to SLO images with the adaptive optics being "off" to most accurately represent our modelling, which deals with the elimination of aberrations [5]. This would be an ideal test to observe our technology's ability to perform in the real world.

5.2 Methods

5.2.1 Collecting Images

To collect the proper images, we plan to recruit 5 subjects. For each subject, we can take 2 images with an SLO and 2 images with an AOSLO (one image for each eye). This would give us 10 pairs of images for direct comparison.

5.2.2 Integrating with the pipeline

We should use a similar pipeline structure used for the evaluation process in Chapter 2. A major difference is relying on authentic pairs for our dataset instead of convolution kernels. The SLO images can be split into small patches that are inputted into our best trained model, using the same splitting process seen in *2.2.5*. Once patches

are processed, we can stitch them (see *2.2.5*). Having matching pairs allows us to run evaluation metrics and also compare cone densities.

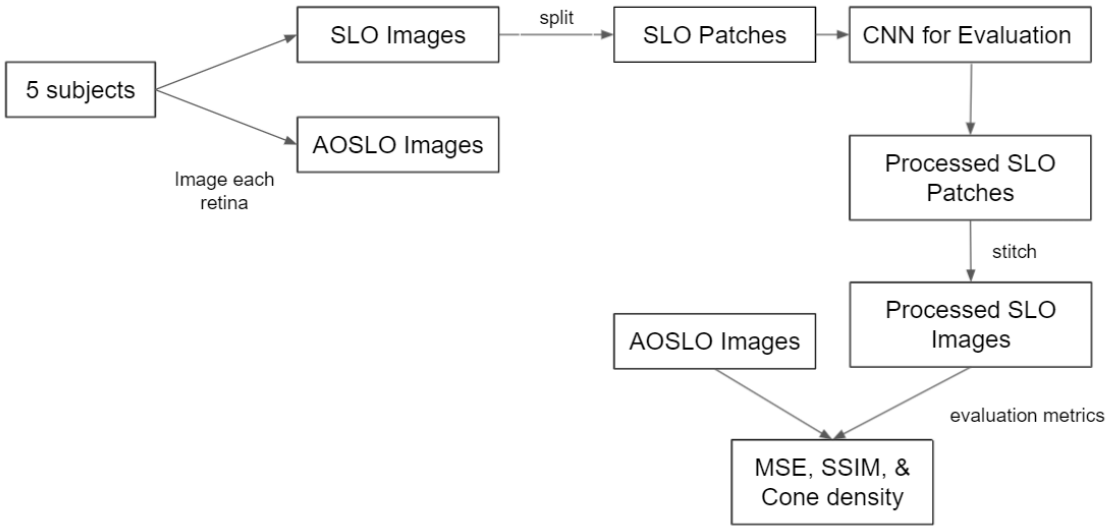

Figure 5.1: Our complete pipeline for using real world data.

5.3 Future Directions

This technology can provide us with the ability to have high quality retinal images without optical aberrations (without needing an AOSLO system). Expensive instruments which need constant maintenance, such as the AOSLO's deformable mirrors, would not be needed in all cases [37]. This can increase accessibility to research labs and physicians without good equipment. The pipeline, including the pre-trained network, can be transformed into an open-source software that can be applied to SLO images to improve their quality. Such a software could be an easy-to-run python script which outputs SLO images with aberration corrections. In addition, using what we have developed so far, we can focus on more intricate challenges within AOSLO images, such as compensating for aberrations for low quality AOSLO images [13].

5.4 Conclusion

Using our preliminary results, we have created a vision to move our project forward, where with proper resources, we could show the usefulness of our software. Although we currently only have a proof of concept, with a proper dataset and test results, we could demonstrate the ability to obtain SLO images with ocular aberrations and remove those aberrations in the post-imaging phase. This could open up a new avenue of using machine learning in ophthalmoscopy to deal with the challenges of cost and availability.

Acknowledgments:

We would like to thank Dr. Yohua Zhang and his team at the UCLA Doheny Eye Institute for providing us with retinal images and research suggestions.

Materials from our previous abstract publication were used in this thesis:

Kassir M, Wang X, Zhang Y, Guo J. Improving image resolution of ophthalmoscopy using artificial neural networks. Investigative Ophthalmology & Visual Science 2021; 62: 2130.

References:

[1] Ganguly D, Chakraborty S, Balitanas M, Kim T. Medical imaging: a review. Communications in Computer Science and Information Science 2010; 78: 504-516.

[2] Zhang Y, Girkin CA, Duncan JL, Roorda A. Adaptive optics scanning laser ophthalmoscopy. Advanced Biophotonics: Tissue Optical Sectioning 2016; Chapter 14.

[3] Litts K, Cooper R, Duncan J, Carrol J. Photoreceptor-based biomarkers in AOSLO retinal imaging. Invest Ophthalmol 2017; 58(6): 255-267.

[4] Browne M, Ghidary S. Convolutional neural networks for image processing: an application in robot vision. Advances in Artificial Intelligence 2003; 2903: 641-652.

[5] Roorda A, Romero-Borja F, Donnelly W, Queener H, Herbert T, Campbell M. Adaptive optics scanning laser ophthalmoscopy. Opt. Express 2002; 10: 405-412.

[6] Fischer J, Otto T, Delori F, Pace L, Staurenghi G. Scanning laser ophthalmoscopy (SLO). High Resolution Imaging in Microscopy and Ophthalmology: New Frontiers in Biomedical Optics 2019; Chapter 2.

[7] Roorda A. Applications of adaptive optics scanning laser Ophthalmoscopy. Optom Vis Sci 2011; 87(4): 260-268.

[8] Booth M. Adaptive optics in microscopy. Philos Trans A Math Phys Eng Sci 2007; 365(1861): 2829-2843.

[9] Talcott KE, Ratnam K, Sundquist SM, Lucero AS, Lujan BJ, Tao W, Porco TC, Roorda A, Duncan JL. Longitudinal study of cone photoreceptors during retinal degeneration and in response to ciliary neurotrophic factor treatment. Invest Ophthalmol Vis Sci 2011; 52(5): 2219-26.

[10] Kim M, Yun J, Cho Y, Shin K, Jang R, Bae HJ, Kim N. Deep learning in medical imaging. Neurospine 2020; 17(2): 471–472.

[11] Zhang Q, Sampani K, Xu M, Cai S, Deng Y, Li H, Sun J, Karniadakis G. AOSLO-net: A deep learning-based method for automatic segmentation of retinal microaneurysms from adaptive optics scanning laser ophthalmoscope images. Image and Video Processing 2021; arXiv:2106.02800.

[12] Ronneberger O, Fischer P, Brox T. U-Net: convolutional networks for biomedical image segmentation. Computer Vision and Pattern Recognition 2015; ArXiv:1505.04597.

[13] Vogel CR, Arathorn DW, Roorda A, Parker A. Retinal motion estimation in adaptive optics scanning laser ophthalmoscopy. Opt. Express 2006; 14: 487-497.

[14] Hofer H, Sredar N, Queener H, Li C, Porter J. Wavefront sensorless adaptive optics ophthalmoscopy in the human eye. Optics express 2011; 19(15): 14160–14171.

[15] Baumal CR. Imaging in diabetic retinopathy. Current Management of Diabetic Retinopathy 2018; Chapter 4.

[16] Lu J, Gu B, Wang X, Zhang Y. High-speed adaptive optics line scan confocal retinal imaging for human eye. PloS one 2017; 12(3): e0169358.

[17] Sekou TB, Hidane M, Olivier J, Cardot H. From patch to image segmentation using fully convolutional neural networks -- application to retinal images. Computer Vision and Pattern Recognition 2019; ArXiv:1904.03892.

[18] Lu J, Gu B, Wang X, Zhang Y. Adaptive optics parallel near-confocal scanning ophthalmoscopy. Opt Lett 2016; 41: 3852-3855.

[19] Burns SA, Elsner AE, Sapoznik KA, Warner RL, Gast TJ. Adaptive optics imaging of the human retina. Prog Retin Eye Res 2019; 68:1-30.

[20] Syed R, Sundquist SM, Ratnam K, Zayit-Soudry S, Zhang Y, Crawford JB, MacDonald IM, Godara P, Rha J, Carroll J, Roorda A, Stepien KE, Duncan JL. High-resolution images of retinal structure in patients with choroideremia. Invest Ophthalmol Vis Sci 2013; 54(2): 950–961.

[21] Lu J, Gu B, Wang X, Zhang Y. High speed adaptive optics ophthalmoscopy with an anamorphic point spread function. Opt Express 2018; 26(11): 14356-14374.

[22] Saito K, Nozato K, Suzuki K, Roorda A, Dubra A, Song H, Hunter JJ, Williams DR, Ethan Rossi EA. Rods and cones imaged with a commercial adaptive optics scanning light ophthalmoscope (AOSLO) prototype. Invest Ophthalmol Vis Sci 2014; 55(13): 1594.

[23] Watson A. Computing human optical point spread functions. J Vis 2015; 15(2): 1-25.

[24] Schwiegerling J. Scaling Zernike expansion coefficients to different pupil sizes. J Opt Soc Am A 2002; 19: 1937-1945.

[25] Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial networks. Machine Learning 2014; arXiv:1406.2661.

[26] He Z, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. Computer Vision and Pattern Recognition 2015; ArXiv:1512.03385.

[27] Kingma DP, Ba J. Adam: a method for stochastic optimization. Machine Learning

2014; ArXiv:1412.6980.

[28] Wang Z, Bovik AC, Sheikh HR, Simoncelli EP. Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing 2004; 13(4): 600-612.

[29] Sara U, Akter M, Uddin MS. Image quality assessment through FSIM, SSIM, MSE, and PSNR - A comparative study. Journal of Computer and Communications 2019; 7(3): 8-18.

[30] Pishro-Nik H. Statistical inference II: Bayesian Inference. Introduction to Probability, Statistics, and Random Processes 2014; Chapter 9.

[31] Wang Z, Simoncelli EP, Bovik AC. Multiscale structural similarity for image quality assessment. The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers 2003; 2: 1398-1402.

[32] Abozaid MA, Langlo CS, Dubis AM, Michaelides M, Tarima S, Carroll J. Reliability and repeatability of cone density measurements in patients with congenital achromatopsia. Adv Exp Med Biol 2016; 854: 277–283.

[33] Zhang F, Qi Z, Duan K, Xi D, Zhu Y, Zhu H, Xiong H, He Q. A comprehensive survey on transfer learning. Machine Learning 2019; ArXiv:1911.02685.

[34] You K, Kou Z, Long M, Wang J. Co-tuning for transfer learning. NeurIPS 2020; 33: 17236-17246.

[35] Gong Z, Zhong P, Hu W. Diversity in machine learning. Computer Vision and Pattern Recognition 2018; ArXiv:1807.01477.

[36] Kinoshita Y, Kiya H. Checkerboard-artifact-free image-enhancement network considering local and global features. Image and Video Processing 2020; ArXiv:2010.12347.

[37] Bartsch D, Mahima J, Zawaydeh Q, Cordeiro MC, Freeman WR. Comparison of adaptive optics scanning laser ophthalmoscopy and high-magnification scanning laser ophthalmoscopy. Invest. Ophthalmol. Vis Sci. 2021; 62(8): 24.