# Adaptive Optics-Scanning Laser Ophthalmoscopy (AO-SLO) Image Analysis Using Deep Learning -- Cone/Rod Recognition

Team Members: Mengxi Zhou, Sruthi Ammannagari, Fan Shen, Zilin Wang

## 1. Introduction

AI has shown its ability in the field of medical domain. In the field of optometry, significant research efforts have been made to present both AI based solutions for identifying several retinal diseases like glaucoma, diabetes, DMR and so on, by screening the retinal images.

Adaptive Optics - Scanning Laser Ophthalmoscopy (AO-SLO) is a popular non-invasive imaging technique which views a distinct layer of the living eye at the microscopic level. It uses confocal methods to diminish extra light by focusing detected light through a small pinhole, which makes possible the imaging of individual layers of the retina with great distinction. Moreover, Scanning Laser Ophthalmoscopy (SLO) itself can be problematic when monitoring individual retinal cells because of optical aberrations created from the tissues of the anterior eye, e.g. cornea. However, the problem can be alleviated by the introduction of Adaptive Optics (AO). AO works by measuring the distortions in a wavefront and compensating for them with a device that corrects those errors such as a deformable mirror or a liquid crystal array. Also, multiple runs of the scanning process with a posterior registration of all scanning results can further improve the image quality by removing the eliminating the effect of eye motions.

The retina has many layers of various cell types. There are two types of photoreceptor cells - *Rods* and *Cones*. These form the outermost layer. These are the photoreceptors responsible for mediating the sense sight. Cones are responsible for the visual acuity of the human eye (the ability of the eye to resolve and to pick up the minor details on an object) and for distinguishing colors. Rods are not able to distinguish colors. However, they are highly sensitive to low-intensity light, thus are mainly responsible for night vision. Cone density measurement is a very sensitive way of detecting changes in the integrity of photoreceptors in eye diseases as well as presence of retinal stretch as in the case of myopia. It is to be expected that the cone density decreases with increasing myopia

The AO-SLO high-resolution images have been used to investigate various changes in the appearance of the photoreceptors in both normal and diseased eyes. Since manual segmentations are subjective, several automated methods were used for detecting cones and rods in the images like Graph based dynamic programming (GDP), Kernel Regression using GDP, Edge learning with GDP, and so on. Deep Convolutional neural networks (CNN) can also be applied to learn directly from the data. In the current work, we have the dataset provided by the College of Optometry, OSU where the centers of cones and rods are marked. With the labeled data set, supervised deep learning methods inspired by the Unet and Unet++ architecture are applied to identify the cones and rods in the dataset and metrics like precision, recall and f1-score, were used to evaluate the results.

# 2. Dataset

The dataset used in this study is collected and labeled by the College of Optometry, OSU. The study includes 12 subjects. From each subject, 7 to 8 AO-SLO images were taken at different locations on the retina. The dataset contains 91 AO-SLO images in total. The labeling process first used a rule-based algorithm to find intensity peaks which potentially serve as cone centers. Then the domain experts manually modified the cone (or rod) centers by adding cones that are missing or by removing those are wrongly identified. The final labeling results will serve as the ground-truth for the learning and validation phase of the deep learning models. A sample AO-SLO image along with its ground-truth labels are shown in Fig 2.1.

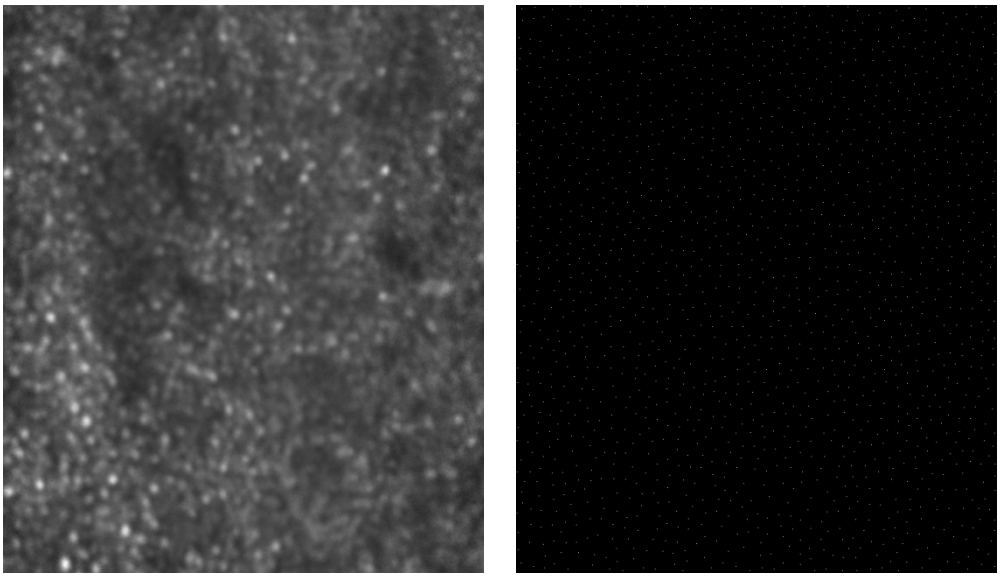

Figure 2.1: Left: a sample AO-SLO image; Right: markers of the cone centers for the left image.

# 3. Proposed Method

In this section, we introduce our proposed method for recognizing the cones and rods in AO-SLO images.

## 3.1. Problem Statement

Domain experts believe that the density of cones and rods in a certain area on the retina can be a key indicator of diseases such as glaucoma. In order to compute the cones/rods density, it is equivalent to count all cones/rods in the AO-SLO image (and then divided by the actual area). Further, it is equivalent to identify all centers. Thus, the problem is defined as the following:

Given an input AO-SLO image I, the problem is to classify each pixel in I as either 0 (not at the center of a cone) or 1 (at the center of a cone).

## 3.2. Workflow

The overall workflow of the proposed method is shown in Figure 3.1.

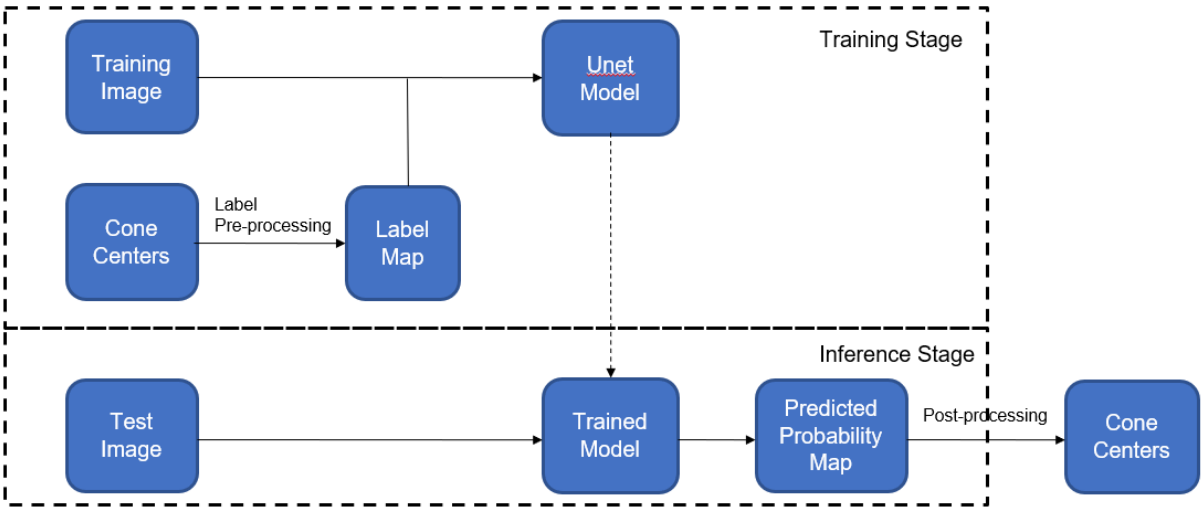


Figure 3.1: Workflow of the proposed method

In the training stage, we first did a transformation on the ground truth labels. This was done in two approaches:

- 1-pixel: this turned the (x, y) coordinates of cone centers into a binary map where the centers were marked as 1s and the background was marked as 0s.
- Expanded: this approach not only marked the center but also the small local area (a circle with the diameter of 10 pixels) around center as 1s. It aimed to alleviate the unbalanced label map where background dominated as opposed to the cone centers.

Then the transformed label map along with the original image was learned by a neural network model (described in section 3.3).

In the inference stage, an unseen image would be fed into the neural network model and the model made prediction over the whole image which generated a probability map. The pixels in the probability map have values between [0, 1] and tell how likely they serve as cone centers. Finally, we did a post-processing step (see section 3.4) that translated the probability map to the (x, y) coordinates of the predicted cone.

## 3.3. Deep Learning Models

In this section, we describe three neural network architectures that we explored in the context of cones and rods recognition.

### 3.3.1. CNNs

David Cunefare, et al. (2017) proposed a method using Convolutional Neural Networks (CNNs) to recognize the cones/rods. Overall, the method divided a whole AO-SLO image into small

patches and tried to predict each small patch as either having a cone or not having a cone. The workflow of the proposed work is shown in Figure 3.2.

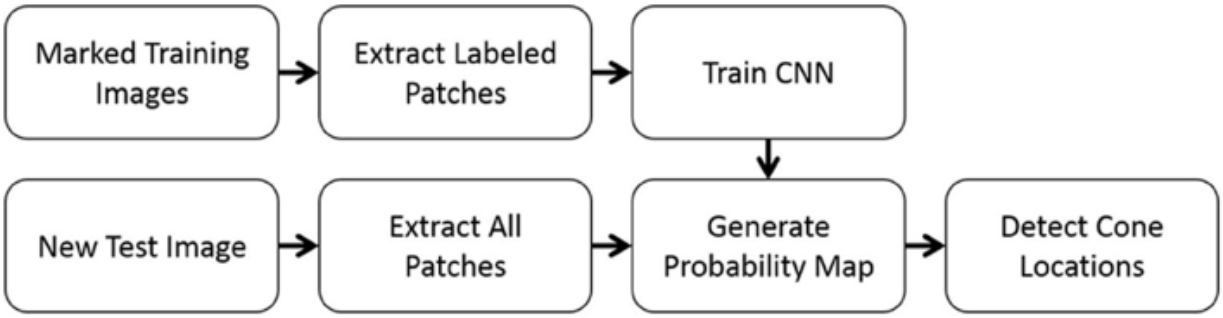


Figure 3.2: Workflow of the method by David Cunefare, et al. (2017)

When doing the training, the method extracted both positive and negative patches according to the ground-truth. For each labeled cone position (x, y coordinates pair), it first extracted a small path around the cone in the image and marked the patch as a positive sample. Further, the authors implemented a simple technique using Voronoi diagrams to extract the non-cone patches and marked them as negative samples. The learning aimed to enable the Convolutional Neural Network (CNN) to classify any given patch as either a cone one or a non-cone one. The architecture of the network can be straightforward, such as vgg19, whose input is an image and the output is a binary label (0 or 1).

When doing inference, for an unseen image, the patches in the image would be extracted by a small sliding window (with the same size as used in the training process) going through the whole image and collecting at each position. The CNN did the inference for each of these patches and turned the prediction scores into a probability map, telling how likely at each position in the image there was a cone. Finally, some post-processing steps, such as Gaussian smoothing, non-maximum suppression, were applied to translate the probability map to actual cone coordinates.

We have reproduced this method in our project which would serve as a baseline comparing to our proposed methods described in the following sections.

### 3.3.2. Unet

Inspired by the U-NET, we adopted a similar architecture here for the cones/rods recognition. The network architecture is illustrated in Figure 3.3. It consists of several compressing encoder blocks followed by several expansive decoder blocks. Skip connections are used to pass the feature representations from encoder blocks to their matched decoder blocks by concatenation. The final decoder output will pass through a classification layer to make the pixel-wise prediction (on whether a pixel is at the center of a cone or not). Each individual block in the architecture are described as follows:

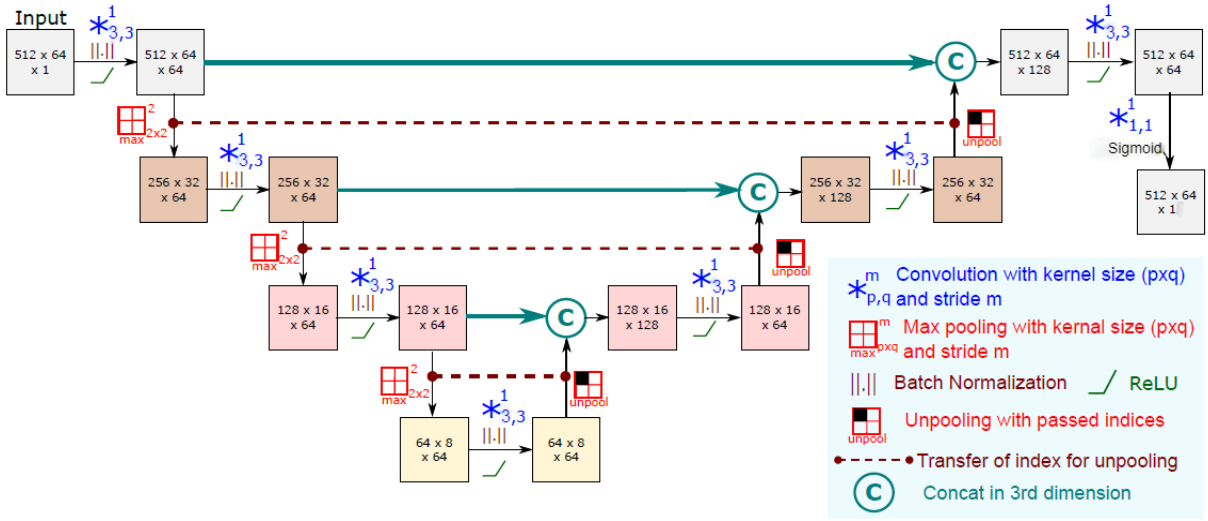


Figure 3.3: Proposed fully convolutional neural network architecture. The spatial resolution of the feature maps indicated in the boxes are just for illustration. Real resolution depends on the input image.

The encoder aims to compress the information in the image into a high-level feature map. As shown in Figure 3.3, each encoder block consists of 4 main layers, in sequence: a convolution layer, a batch normalization layer, an activation layer (ReLU) and a max pooling layer. The kernel size is 3×3 for all convolutional layers in the encoder and the stride is 1×1. A zero-padding is applied to all convolutional layers so that the dimension before and after the convolution layer is kept. The number of channels for the output feature map is 64 for all convolutional layers. Batch normalization layers are added to help mitigate the covariate shifts problem and speed up the training procedure. We use ReLU as the activation function which introduces non-linearity. This is followed by a max-pooling layer. The filter size and the stride are both 2×2 for the max-pooling layer, thus reducing the feature map dimensions by half. Also, the pooling indices in the max-pooling layers are stored and used in the corresponding unpooling stage of the decoder block. Rather than doing the unpooling in a simple duplication manner or in an interpolation manner, this provides more spatial consistency. An explanation of the usage of pooling indices is provided in Figure 3.4.

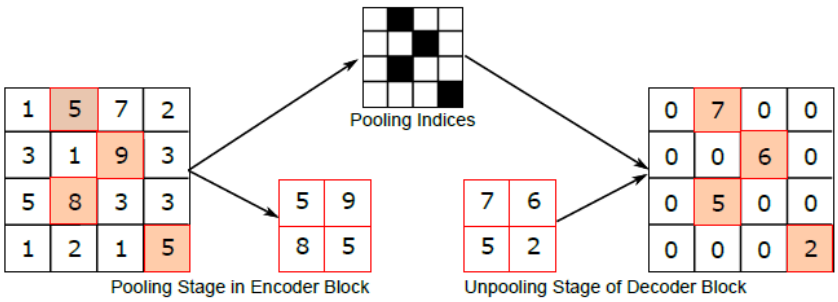


Figure 3.4: Illustration of pooling and unpooling procedures. Pooling indices are stored in the pooling stage and are leveraged in the unpooling stage preserving appropriate spatial locations.

The decoder untangles the information in the final feature map generated by the encoder. It captures the key information (i.e. where the cones are) and restores it into a feature map with the same size as the input image. There are 3 decoder blocks. Each of them consists of 5 layers, in sequence: an unpooling layer, a concatenation layer, a convolution layer, a batch normalization layer and an activation layer (ReLU). The unpooling layer up-samples a coarse feature map from the preceding decoder block to a finer resolution by using saved pooling indices from the matched encoder block and imputes zeros at the rest of the locations. A concatenation layer, usually called skip connection, follows the unpooling layer. It broadcasts the information from the matched encoder block to the current unpooled feature map by doing a concatenation (on the last dimension of the feature maps). Such a skip connection helps the transition to a finer resolution by adding an information rich feature map from the encoder part. Additionally, it helps avoid the problem of gradient vanishing. The convolutional layers have the same filter sizes and strides as the encoder part, i.e. 3×3 and 1×1. Zero-padding is again applied to keep the appropriate size of the feature map. The convolutional layer helps densify the sparsely unpooled feature maps from the previous layers. Additionally, a batch normalization layer and an activation are added, which serve the similar effect as they are in the encoder part.

The final decoder block is followed by a convolutional layer with 1*1 kernels (used for reducing channels of the feature map without changing spatial dimensions) to map the 64 channel feature map to a single channel feature map (for a binary classification). Sigmoid function is used for this layer which makes sure the output is in range [0, 1].

### 3.3.3. Unet++

We also adopted a modified version of U-NET, called U-NET++, for our cones/rods recognition task. This model architecture was proposed by Zhou etc. from Arizona State University and has been proven to have better performance on a number of datasets. The general U-NET++ model architecture is shown in the Figure 3.5 below. As the name suggests, U-NET++ share almost the same encoder/decoder architecture except from the two aspects that will be introduced in the following paragraphs:

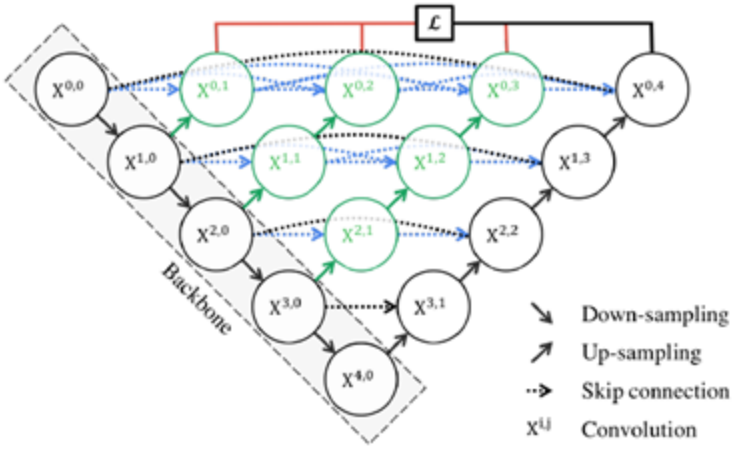


Figure 3.5: U-NET++ Architecture

Firstly, the skip pathways in U-NET++ are redesigned to have smoother and more reasonable information flow. As we can see, Figure 3.6 serves as an example of how the top skip pathway is formed. Specifically, the new skip pathways include convolutional layers followed by an activation layer. The capital letter **H** in Figure 3.6 exactly denotes the new convolute then activate operation mentioned above. The main benefit from this part is that the semantic gap between the encoder sub-network and decoder sub-network can be effectively fulfilled. Another important part is that the new skip pathways have nested and dense skip connections. By nested and dense connections, the information in feature maps and gradient can flow more properly in the model, which helps the model capture more subtle and fine-grained details. This is a crucial advantage to our cones/rods recognition. Similar to the concatenation layer in the original U-NET model, feature maps from the previous part of the same dense block level are concatenated (denoted by []) with the up-sampled feature maps (denoted by **U**) from the lower dense block level. The ideas behind these modifications are from DenseNet.

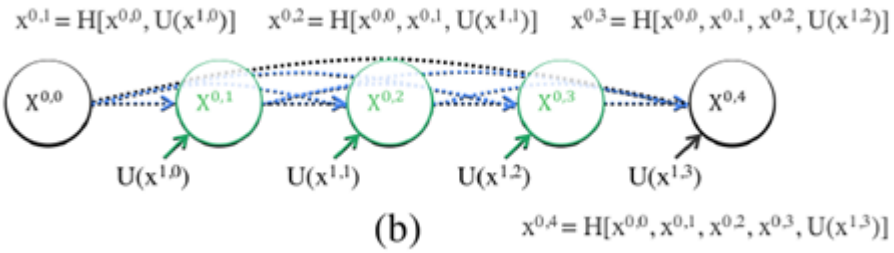


Figure 3.6: Re-designed Skip Pathways

Secondly, in the U-NET++ architecture, researchers have the option to incorporate deep supervision in the model. The deep supervision part is shown on the top of Figure 3.5 by the red line. With deep supervision, the model is able to combine the feature maps from different levels in two modes: accurate mode and fast mode. In short, when operating in accurate mode, the model takes the average of all feature maps, whereas when operating in fast mode, the model

only chooses one set of feature maps as the final set of feature maps. The main benefit of incorporating deep supervision is that it can act as the regularization measure, which improves the convergence performance.

## 3.4. Post-processing

The final output of the neural network model is a map having the same size as the input image. Each pixel in the map has a value in the range of [0, 1], indicating how likely it is at the center of a cone/rod. Further we did certain post-processing steps to binarize the decision so that the prediction is straightforward to understand and comparable to the ground-truth.

### 3.4.1. Non-maximum Suppression

Given a simple fact that within each cone there should only be one center, we did a non-maximum suppression step to avoid more than one pixel being predicted as a center. In a small local area (20*20 in our experiments) around one pixel, the pixel only retained its prediction value if the value is the highest. Otherwise this pixel's value will be set to 0. The size of the local area can further be defined by the domain knowledge, i.e. how large a cone usually is in a certain AO-SLO imaging setting. By doing this suppression, the results become easier for people to view and more comparable to the ground-truth markers.

### 3.4.2. Thresholding

Secondly, we did a thresholding on each pixel. For a threshold α, a pixel would be set to 0 if its value is less than α. Otherwise it would be set to 1. This step simply binarized the prediction and enabled the comparison between a prediction map and the corresponding ground-truth map.

### 3.4.3. Ground-truth Matching

Finally, we did a matching between positively predicted pixels and ground-truth markers. The reason for this type of matching is because a positively predicted pixel may sit right next to a ground-truth marker but not exactly overlap with the marker. Since our final goal is to count the number of cone centers, as long as one point is captured within a cone, the goal can still be achieved. Therefore, in such a situation, we thought the positively predicted pixel should be considered as correctly predicted. We gave a small allowance (a 10-pixel distance in our experiments) for a positively predicted pixel to match to a nearby ground-truth marker. This allowance is also a hyperparameter in the procedure and can benefit from domain knowledge, e.g. how large a cone usually is, for better cone counting results.

# 4. Experiments and Results

In this section, we describe our experiment settings and discuss the results.

## 4.1. Experiment Settings

We first divided the whole dataset into a training set and a validation set. Since currently the dataset is small, we didn't set aside a portion for the test set. The training set consists of 74

images and the validation set has 17 images. Certain data augmentation techniques (i.e. image shifting, flipping) were used to generate more training samples.

We developed our model in the Keras framework. The model was trained for 150 epochs in total. Adam optimizer was used with an initial learning rate of 0.1. And the learning rate was reduced by an order of magnitude after every 30 epochs. We used the binary cross entropy as the objective function for the neural network.

## 4.2. Results

Standard metrics, i.e. precision, recall and f1-score, were used to evaluate the results. Figure 4.1 shows an example AO-SLO image with the predicted cone centers marked in the image. The color coding in the image is as follows:

- Green: These are the true positives, meaning the model predicted there is a cone center and it actually is. Remember in the post-processing, we give a small allowance for a predicted pixel to match to a marker in the ground-truth.
- Yellow: These are the false positives, meaning the model predicted there is a cone center but there should not be one.
- Red: These are the false negatives, meaning there should be a cone center but the model did not capture it (even if the allowance is given).

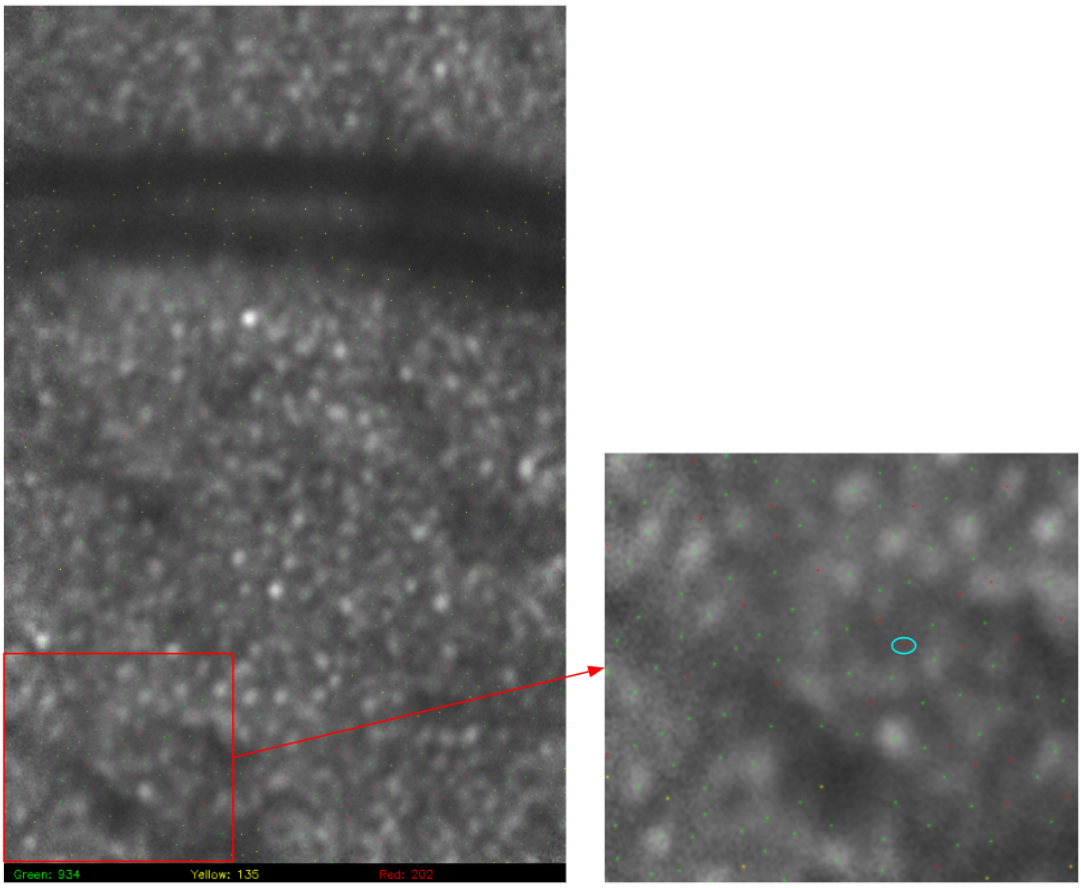


Figure 4.1: An AO-SLO image with the predictions marked in it. A close-up of the left bottom of the image is shown on the left.

The threshold used to generate Figure 4.1 is 0.2. In general, the proposed method is promising as the green dots (true positives) dominate. However, the drawbacks are obvious. For example, in the black band area (which represents the shade of the blood vessel), numerous false positive predictions were made. Also, the precision (=0.87) and recall (=0.82) scores are not satisfactory. We are seeking further improvements by exploring the hyperparameter space and generating more training data.

Another noticeable fact is our ground-truth labels are not perfect. For instance, see the red dot highlighted by a cyan circle in Figure 4.1. The ground-truth marked a cone center at the red dot, but domain experts verified that there should not be one. Thus, a further refinement on the ground-truth labels is needed.

Furthermore, we show a comparison between different neural network architectures that we have implemented in Table 4.1. The version number “v0” indicates the model is according to the original design of the model. “v1” indicates that we have made modifications to the model, such as the up-pooling mechanisms, number of filters in the convolutional layers, etc.

| Model_name | precision | recall | f1 | loss | pre-processing |
|---|---|---|---|---|---|
| Unet_v0 | 0.7884 | 0.8557 | 0.8207 | bce | 1-pixel |
| Unet_v1 | 0.8182 | 0.8178 | 0.8180 | bce | 1-pixel |
| Unet_v1 | 0.8084 | 0.8353 | 0.8216 | bce | expanded |
| Unet++_v0 | 0.7916 | 0.8755 | 0.8314 | bce | 1-pixel |
| Unet++_v1 | 0.7739 | 0.9034 | 0.8337 | bce | 1-pixel |
| Unet++_v1 | 0.734 | 0.8861 | 0.8029 | focal | 1-pixel |
| Unet++_v1 | 0.8482 | 0.8241 | 0.8360 | bce | expanded |
| CNN (David) | 0.8160 | 0.9374 | **0.8725** | bce | - |

Table 4.1: Comparison of different models’ performance.

# 5. Another Approach: Object Detection

We also experimented on preliminary works towards using existing object detection algorithms that utilizes bounding boxes to recognize objects within a certain image. We were able to conduct pre-processing work on our dataset based on the provided ground truth data. The ground truth data consists of expert manually selected centers for cons and rods in our scan and their corresponding coordinate data in respective images.

Utilizing an open source adaptation of xarray, rioxarray, we were about to extract all the pixel intensity data along with their coordinate data so that we can import the ground truth dataset and find starting points for the bounding boxes. Once the starting point is found, the pixel intensity of the adjacent pixels of the starting point will be compared to the starting point. The subtraction of pixel intensity between the subject pixels and starting pixel is then used as a parameter to determine the boundary where the high intensity cons and rods area transitions to the low intensity background areas. Initially, the pixel difference parameter was set to a fixed value such as 80 so that whenever a pixel was found to be 80 lighter than the center starting point, the boundary will be set. We also set a second parameter for the maximum steps that our algorithm can take so that the bounding box can have an upper limit size, as some of the cons and rods have so little contrast against the background that the difference never reaches the preset parameter. Shown in fig 5.1. is the first iteration with fixed difference parameter:

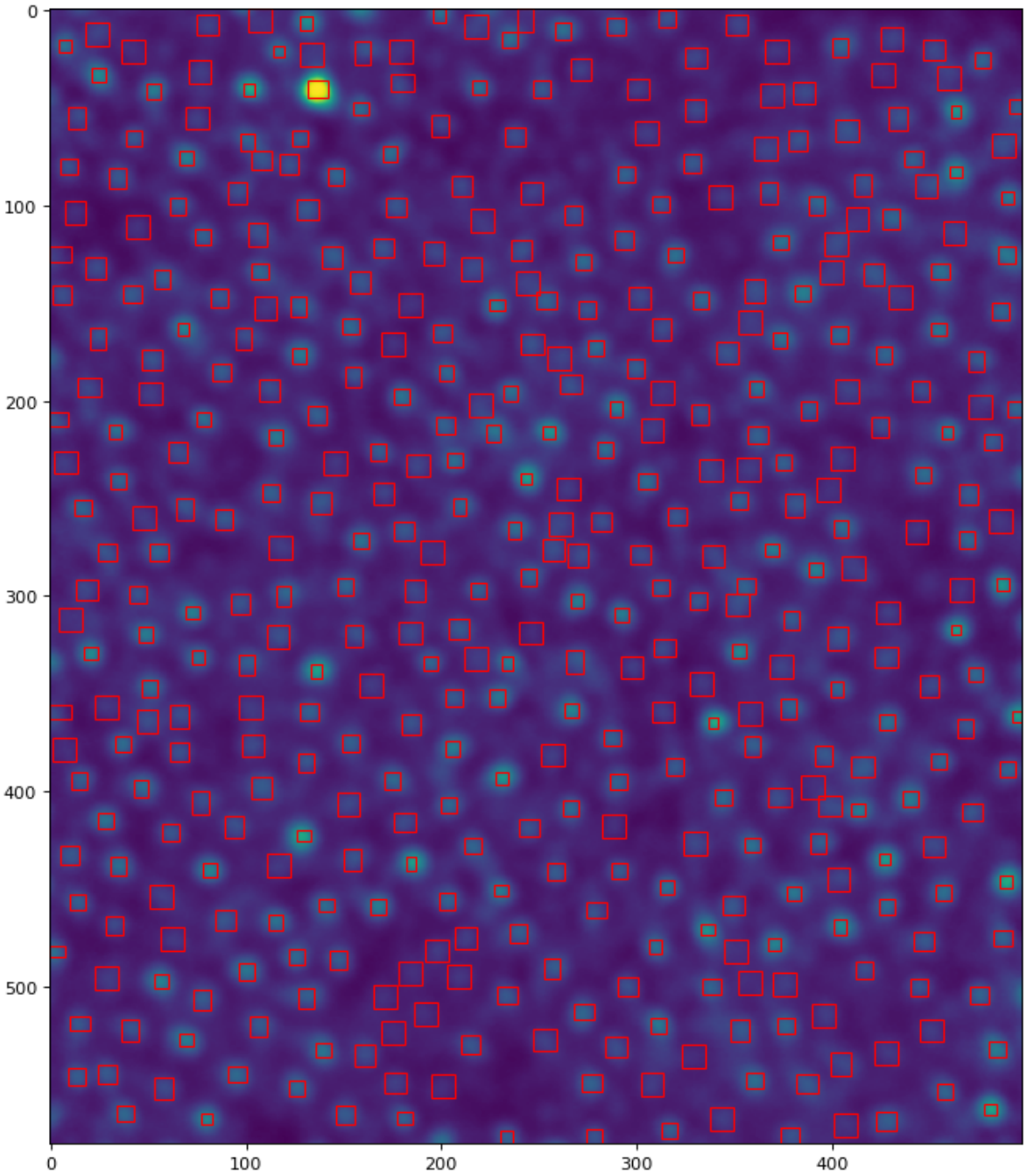


Figure 5.1: Bounding boxes with difference parameter = 50 and size parameter = 10

The first iteration showed multiple problems including that the difference parameter being too small for the high contrast cons and rods that the bounding box only covers the very center of them. The max size is also too small to cover the lower contrasted ones. A second revised image with higher tolerances was then produced.

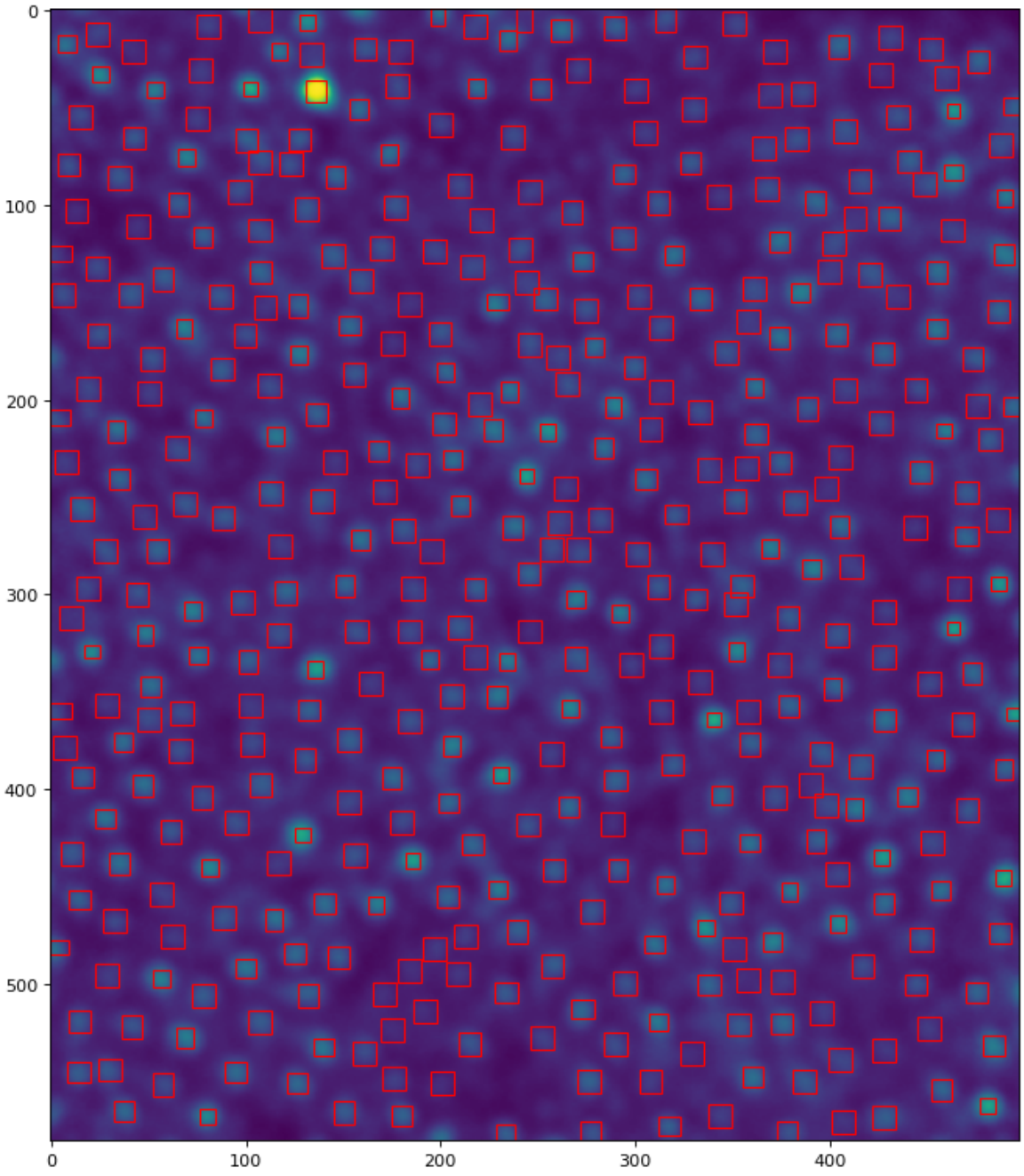


Figure 5.2: Bounding boxes with difference parameter = 80 and size parameter = 12

The higher tolerances contributed little difference in the overall coverage of cons and rods in the original dataset, so an advised algorithm was proposed: instead of using a fixed variable difference parameter that one-size-fits-all for all cons and rods in the dataset, we opted for a percentage based difference parameter. The new parameter instead of using a fixed comparison value, calculates the percentage difference between the subject pixel and the center pixel, making it less likely for the algorithm to

hit the upper limit while processing lower contrast subjects while providing more complete coverage for high contrast subjects.

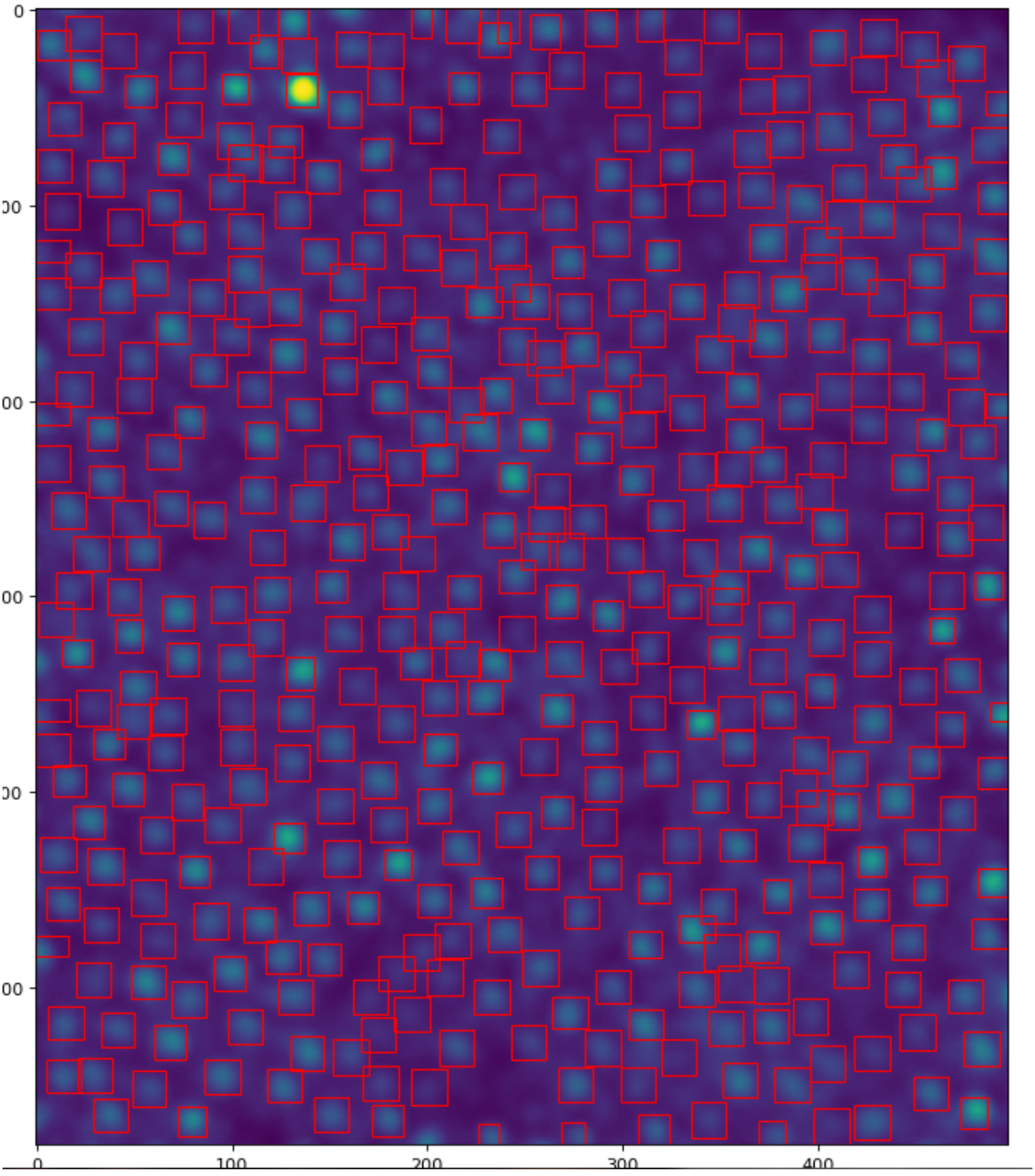


Figure 5.2: Bounding boxes with difference parameter = 55% and size parameter = 16

The preprocessing will be applied to all of the AL-SLO scans to prepare them for existing object recognition algorithms for training and testing in the future.

# 6. Future Work

For the future work, we aim to make improvements on the followings:

- Ground-truth refinement: this needs the help from the domain experts that they need to further verify or revise the current ground-truth. A more accurate ground-truth certainly will improve the model performance.
- Object detection models: we would like to investigate into several object detection models, e.g. YOLO, Faster-RCNN, and compare the results with our existing methods.
- Model exploration: for the proposed methods, we further aim to do more explorations, such as hyperparameter tuning.
- Distinguish cones and rods: another useful aspect will be distinguishing cones and rods when recognizing. This will provide more detailed information on the patient's eye.
- User study: we plan to apply the trained model to the current analysis loop for domain experts. And compare how much this will help the experts do the analysis both quantitively (e.g. time saving) and qualitatively (e.g. survey).