**Read Me File for the paper:**

### David Cunefare, Leyuan Fang, Robert F. Cooper, Alfredo Dubra, Joseph Carroll, Sina Farsiu,

### "Open source software for automatic detection of cone photoreceptors in adaptive optics ophthalmoscopy using convolutional neural networks"

### 

###### *Scientific Reports,* 7, 6620, 2017

**Notes:**

* The code was tested in MatLab 2016b and utilizes MatConvNet-1.0-Beta23 developed by the MatConvNet Team (see http://www.vlfeat.org/matconvnet/).
* Coordinates are saved as (x , y) pixel positions (in Matlab’s default coordinate system) column wise.
* AFLD cords are saved in the “CombinedPos” variable in each .mat file.
* Results of the CNN method are saved in the “Images and Results\...\Validation CNN Coord” folder. Cone coordinates are saved as the “CNNPos” variable in each .mat file.
* Example code for displaying an image marked with cone locations found using a trained network is given in “DisplayMarkedImageExample.m”.

**Running:**

Before running make sure MatConvNet is compiled for gpu (see <http://www.vlfeat.org/matconvnet/install/>)

“vl\_setupnn.m “ should be run to add MatConvNet paths to MatLab’s search path and to check for problems. This is done automatically in the main scripts (“RunCNNConeDetection.m”, “RunCNNnewSet.m”, “DisplayMarkedImageExample.m”). If you have MatConvNet installed to a different folder than the default, change the “MatConvNetPath” variable near the top of the scripts to your path for MatConvNet.

**C-CNN and SD-CNN:**

* To run from the beginning training new network weights, you must first delete or rename the “Images and Results\...\CNN Training-Split Detector” or “Images and Results\...\CNN Training-Confocal folder”, or change the “params.CNN.TrainExpDir” parameter in “get\_parameters\_Cone\_CNN.m” for the specific data set. The “cnn\_Cones.m” function is designed to begin where it has previously left off.

1) Main script is “Code\RunCNNConeDetection.m”

2) To choose between the two provided data sets, change the “DataSet” variable to be confocal, and split detector.

3) Run the script (description of individual functions below)

Results (Sensitivity, False Discovery Rate, and Dice’s coefficient) in comparison to manual grading across the validation data set can be generated by running “FindSplitDetectorResults.m” or “FindConfocalResults.m”. This can be done on the Saved coordinates provided initially.

**M-CNN**

Code for producing the results of the M-CNN is provided in “RunCNNConeDetection\_Combined”, and “get\_parameters\_Cone\_CNN\_Combined.m”.

**New Data Sets:**

1) To use your own dataset for training a network add a new case in the “get\_parameters\_Cone\_CNN.m” following the same formatting. If your data is saved in a different format, you will need to change the way the data is read in the supporting functions.

* Training images should be saved in their own folder. Currently images are required to be readable using matlab’s *imread* function, and to be 2D/grayscale.
* Manual coordinates for the training images should have the same base name as the images (with additional text at the end if wanted). The C\coordinates should saved as (x , y) pixel positions (in Matlab’s default coordinate system) column wise. The program is currently only set up to read .csv or .txt files (in the same format as the data sets provided).
* Validation images and coordinates should follow the same guidelines, but are not required for training a network and finding detection parameters.

2) To run, open “Code\RunCNNConeDetection.m”, change the ““DataSet” variable to the name of your new case and run the script. If you are not using a validation data set, change the “ValidateFlag” variable to 0 and comment out the “SaveValidationCones” function.

To use an already trained network on a new set of images, use “RunCNNnewSet.m”. Change the “DataSet”, “ImageDir”, “ImExtension”, and “SaveDir” variables as explained in the script. (Note: only saves the coordinates of the cone locations)

**New Network Architecture:**

1) Create a new initialization file (following the format of “cnn\_Cones\_init.m”)

2) In (“cnn\_Cones.m”) add a new case for the model (line 38) calling this initialization

3) Change parameters in “get\_parameters\_Cone\_CNN.m”. Change “params.CNNtrain.modeltype” to the new model case, and “params.PatchSize” to the input patch size. Add a new DataSet case or adjust the parameters so that previous results are not saved over.

4) Run “RunCNNConeDetection.m””.

**Main Functions:**

*RunCNNConeDetection*: Main script for training a network, optimizing detection parameters, and saving validation results (cone positions and probability maps)

*RunCNNnewSet*: Script for using a trained network/parameters for detecting and saving cone locations on a new set of images

*get\_parameters\_Cone\_CNN*: Returns hyper-parameters for training network, and directory locations for saving/loading data

*CreateConeIMDB*: Creates an image database of labeled cone and non-cone patches from the training data set based on manual markings.

*cnn\_Cones* and *cnn\_train\_Cones*: Trains the network weights using the extracted image patches

*SaveProbabilityMaps*: Use trained network to create and save probability maps for the training and validation images (TrainFlag and ValidateFlag can be set to determine which sets are processed)

*OptomizeConeDetectionParameters*: Uses a brute fource search to find detection parameters that maximize the average Dice’s coefficient with respect to manual grading over the training data set.

*SaveValidationCones*: Uses trained network and parameters to detect and save cone locations in validation data set

*cnn\_Cones\_init*: function for initializing the CNN architecture and hyper-parameters.

*GetConePosSingle*: function to use a trained network/parameters to find cone positions in a single input image