1. Eye Detection Algorithm

1.1 Introduction

Eye detection is one of the challenging problem in Computer Vision world. There are many techniques for eye detection such that template matching, morphological operations, using Hough/Haar transform to locate iris/pupil of an eye. This problem can be solved using Machine Learning techniques like Support Vector Machine (SVM), Clustering and Neural Network etc.

In the algorithm described here, we first generate all possible Eye Candidates purely based on the fact of corner points. Corners in image are detected using FAST algorithm.
Once we get set of Eye Candidates, we pass it to Pattern Recognition Neural Network which recognizes whether the given Eye Candidate is Left Eye, Right Eye or Not an Eye.

1.2 Algorithm

% DESCRIPTION: Algorithm to detect Left and Right Eye.
% INPUT: RGB image
% OUTPUT: X and Y coordinates of Left and Right Eye.

---

1. IF trained Pattern Recognition Network does not exist,
   Then,
   a. Prepare training data.
      Note: For the details of preprocessing steps applied on training image, please refer section 1.3
   b. Create a Pattern Recognition Network.
      Note: For the details of Pattern Recognition Network, please refer section 1.4
   c. Train Pattern Recognition Network using prepared training data.
   ELSE,
   a. Load trained Pattern Recognition Network.
   ENDIF
2. Find width and height of given input RGB image.
3. Transform image from RGB color space to LAB color space.
   Reason:
   Eye detection is efficient in LAB color model compare to RGM color model.
4. Use 'L' component of LAB image for further processing, where 'L' stands for lightness contrast.
   Reason:
   As Eye dot corresponds to brightest spot in face image, hence using 'L' component in further algorithm returns better results.
5. Detect corner points in image using FAST algorithm.
6. Pick first 50 strongest corner points as possible Eye Candidates.
7. FOR every Eye Candidate (as specified by 'EyeCandidates'),
   a. Get X and Y coordinates of Eye Candidate under inspection.
   b. Crop proportional region surrounding Eye Candidate location,
      Where,
      - X1: X coordinate of top-left corner of image segment to be cropped, i.e. X - (Width/10)

- Y1: X coordinate of top-left corner of image segment to be cropped, i.e. Y - (Height/20)
- Width/5: Width of image segment to be cropped
- Height/10: Height of image segment to be cropped

c. Preprocess cropped Eye Candidate image segment.

Note: For the details of preprocessing steps applied on image segment, please refer section 1.3

d. Feed preprocessed Eye Candidate image segment to Pattern Recognition Network and predict whether it is Left Eye, Right Eye or Not an Eye.
e. Find which output class among 3 (i.e. Left Eye, Right Eye and Not an Eye) has highest probability and accordingly record corresponding output class.
f. Store predicted Left Eye and Right Eye probabilities for Eye Candidate image segment under inspection.

ENDFOR

Note: At the end of step, there can be multiple Left Eye or Right Eye output class predictions. So, next step will pick exact Left Eye and Right Eye candidate among many possible candidates.

8. Locate Left Eye using below steps,
   a. First check which Eye Candidate predicted as a 'Left Eye' has highest Left Eye probability and mark it as final expected Left Eye Candidate.
   b. IF no Left Eye Candidate is found,
      Then,
      I. Pick an Eye Candidate having highest Left Eye probability among all possible Eye Candidates (even if it is not predicted as 'Left Eye') and mark it as final expected Left Eye Candidate.
      ENDIF
   c. IF still no Left Eye Candidate is found,
      Then,
      I. Set default location for Left Eye Candidate.
      ENDIF
9. Locate Right Eye using below steps,
   a. First check which Eye Candidate predicted as a 'Right Eye' has highest Right Eye probability and mark it as final expected Right Eye Candidate.
   b. IF no Right Eye Candidate is found,
      Then,
      II. Pick an Eye Candidate having highest Right Eye probability among all possible Eye Candidates (even if it is not predicted as 'Right Eye') and mark it as final expected Right Eye Candidate.
      ENDIF
   c. IF still no Right Eye Candidate is found,
      Then,
      II. Set default location for Right Eye Candidate.
      ENDIF
10. Set X and Y coordinates for Left Eye and Right Eye using locations found as described above.
11. Stop.

1.3 Image Preprocessing for Neural Network

Training data and Eye Candidate segments are preprocessed before feeding it to Neural Network.

Special Note:

- Training data: If you want to train Neural Network model on training data, please download 'Train' folder from GitHub and copy it to location where your script is running.
- Location: https://github.com/chetanborse007/EyeDetection

Preprocessing steps are as below:

1. Transform image from RGB color space into HSV color space.
2. Use 'V' component for further processing.
   Reason:
   a. 'V' component of HSV image provides a chromatic notion of the intensity of the color, shortly brightness of color.
   b. Eye dot in face image corresponds to bright spot due to light reflecting from it.
   c. Hence, using 'V' component for further analysis ensures better results.
3. Normalize 'V' component of HSV image.
4. Resize 'V' component of HSV image to standard size i.e. 200x100.

1.4 Pattern Recognition Neural Network

This algorithm uses Pattern Recognition Neural Network which recognizes whether the given Eye Candidate segment is Left Eye, Right Eye or Not an Eye. Pattern Recognition Neural Network is trained using training data for Left Eye, Right Eye and Not an Eye.

1.4.1 Neural Network Architecture

Pattern Recognition Neural Network used in this algorithm has one Input Layer (input size [20000]), five Hidden Layers (20, 40, 80, 40, 20 activation units respectively) and one output layer (output class [3]).

Splitting training set:

Training data used by this architecture is split into Training, Validation and Testing sets as below:

```
PatternNet.divideFcn     = 'dividerand';
PatternNet.divideMode = 'sample';
PatternNet.divideParam.trainRatio = 75/100;
PatternNet.divideParam.valRatio     = 20/100;
PatternNet.divideParam.testRatio   = 5/100;
```

Training function:

Here, we are using Scaled Conjugate Gradient backpropagation.

```
PatternNet.trainFcn = 'trainscg';
```

Performance check:

For checking performance of neural network, crossentropy technique is used.

PatternNet.performFcn = 'crossentropy';

Tuning parameters:

1. max_fail: Maximum validation failures (Here, it is set to 8).
2. epochs: Maximum number of epochs to train (Here, it is set to 100).

Regularization:

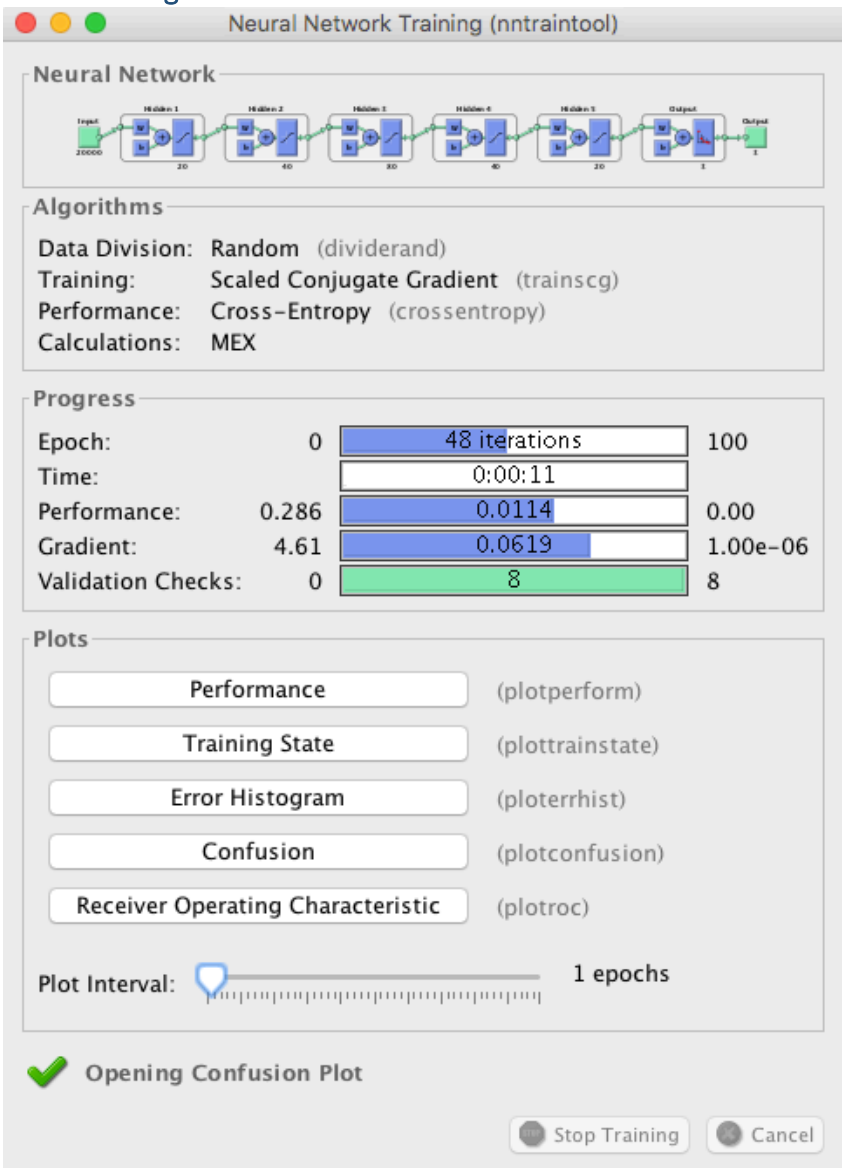
Regularization parameter is set for generalizing Pattern Recognition Network and for avoiding overfitting.

PatternNet.performParam.regularization = 0.5;

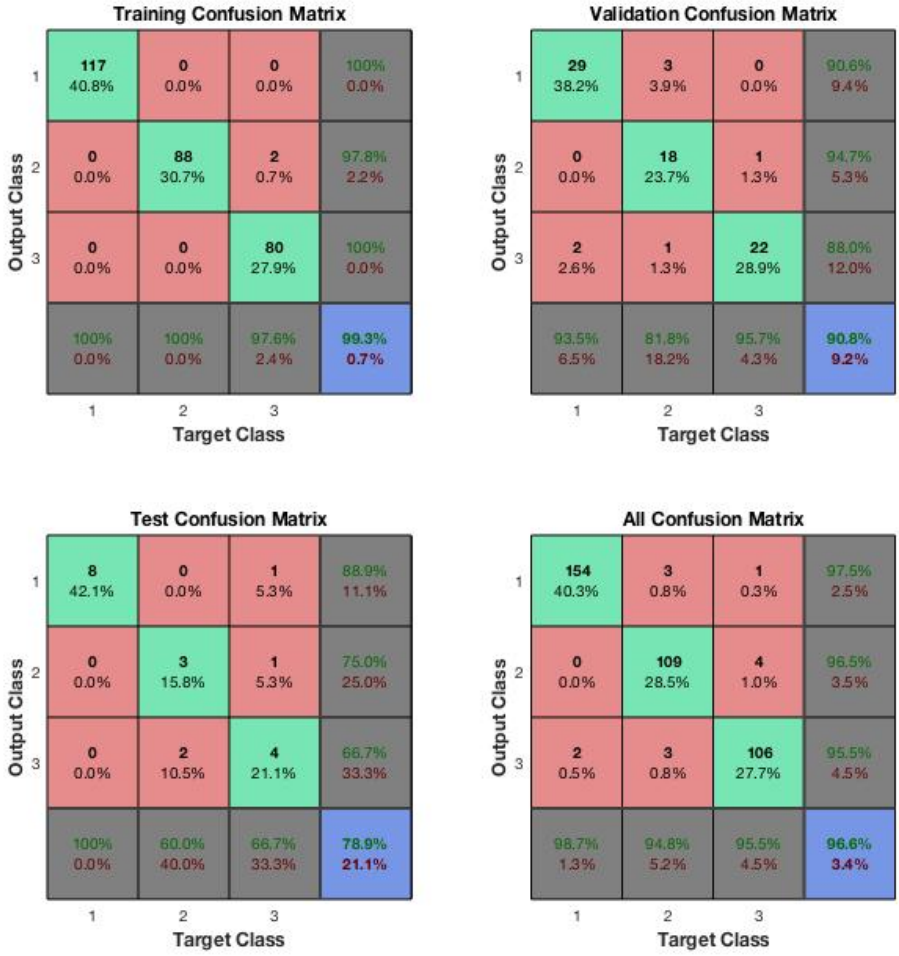
2. Experiments

2.1 Training Neural Network

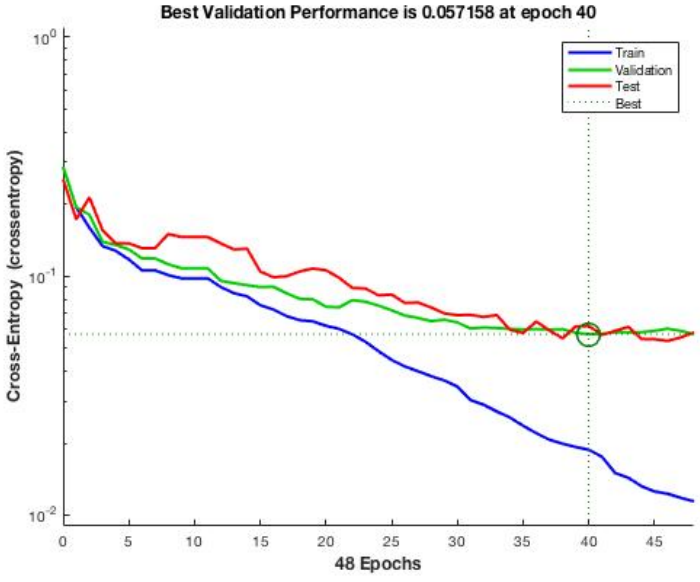
2.1.1 Training statistics



2.1.2 Confusion Matrix



2.1.3 Performance



2.2 Performance of algorithm on validation data

Validation Image	Distance between ground truth
vo1.jpg	0.2180 + 0.0000i
vo2.jpg	0.0128 + 0.0000i
vo4.jpg	0.0184 + 0.0000i
vo5.jpg	0.0249 + 0.0000i
vo6.jpg	0.0119 + 0.0034i
vo7.jpg	0.3538 + 0.0000i
vo8.jpg	0.0165 + 0.0005i
vo9.jpg	0.0060 + 0.0000i
vo10.jpg	0.2460 + 0.0307i

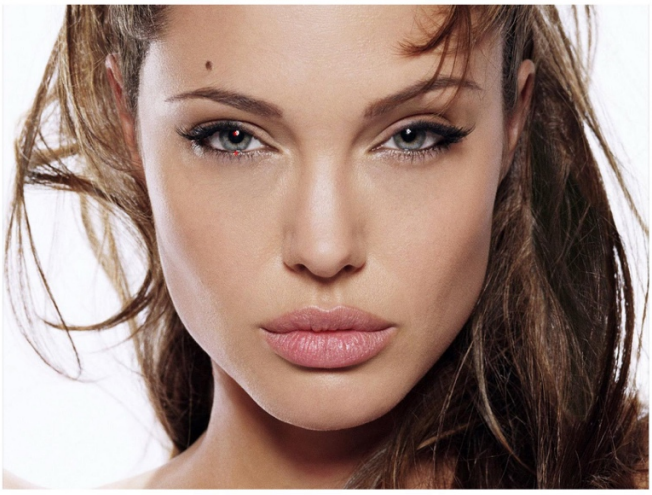
Note: Overall performance looks good except few outliers. Please refer 'Issues and *Improvements'* section for reason behind the outliers.

2.3 Visuals of Eye Detection

3. Issues and Improvements

3.1 More training data

Current training set has about total 350 images for Left Eye, Right Eye and Not an Eye categories, which is not enough. Hence due to this issue, sometimes it predicts Left eye as Right Eye and Right Eye as Left Eye.
e.g.



For better predictions, more training data should be provided to Neural Network.

3.2 Eyes at different orientations and depths

If eye is at certain angle, this algorithm partially fails to predict correct eye location.
e.g.

So, the solution is that training images at different orientations and depths should be provided to Neural Network. Such training data can be generated by applying rotation, translation etc. on existing training images.

3.3 Corner points outside face

There is a chance that algorithm can treat corner point outside a face as probable Eye Candidate.

Such outliers can be avoided using following steps:

- Detect skin using skin segmentation technique, wherein skin ~ face region.
- Crop face region from original image according to segmented skin and treat it as a separate image segment.
- Then, apply FAST algorithm to recognize corner points on cropped face region segment.

3.4 Use Hough/Haar instead FAST

Hough/Haar transform can be used instead FAST algorithm to extract all possible Eye Candidates from face image.

4. References

1. https://en.wikipedia.org/wiki/Lab_color_space
2. https://www.kirupa.com/design/little_about_color_hsv_rgb.htm
3. http://www.mathworks.com/help/vision/ref/detectfastfeatures.html
4. https://arxiv.org/pdf/1205.5097.pdf
5. http://www.mathworks.com/help/nnet/ug/improve-neural-network-generalization-and-avoid-overfitting.html
6. ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia353_1s07/papers/moller_90.pdf
7. http://www.ijser.org/paper/Pattern-Recognition-Neural-Network-for-Improving-the-Performance-of-Iris-Recognition-System.html
8. http://jips-k.org/file/down?pn=259

5. Repository

https://github.com/chetanborse007/EyeDetection