

Intelligent Visual Computing [Spring 2018]

Assignment 2: A Neural Network for Eye Detection

Overview

In this assignment you will learn the basics of deep learning. You will implement a neural network that detects eyes in images. This assignment counts for **15 points** towards your final grade.

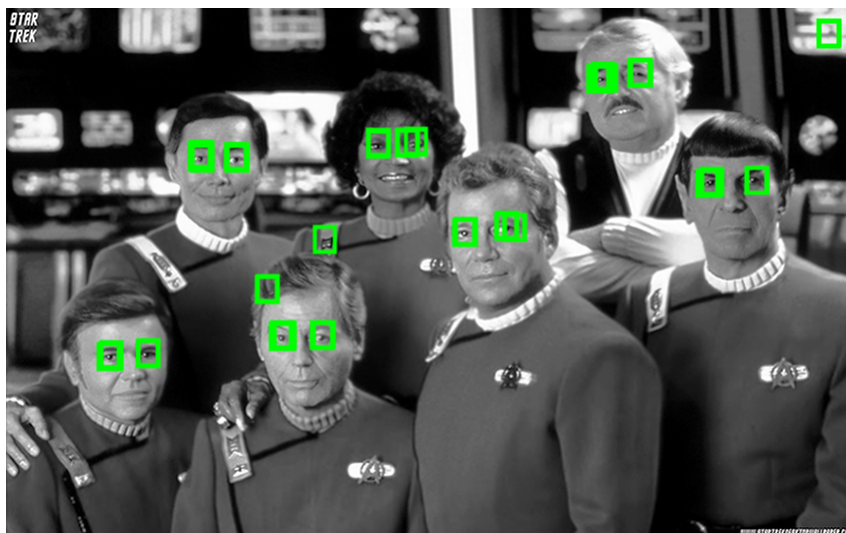


Figure 1: Example of eye detection using the neural network you will implement in this assignment. Green rectangles indicate detected eyes. Your results may slightly vary. It is ok if results are not perfect (e.g. in this example, you may lose 1-3 eyes, or get 4-5 regions mistaken for eyes).

Instructions

The starter code for this assignment is written in Matlab. Download the code here: [starter_code.zip](#). The code has the following scripts and data:

- **trainSet.mat** and **testSet.mat**: a set of eye and non-eye images has been split into two disjoint sets for training and testing. You will use the former to train your eye detector and, once the training is complete, you will use the latter to test its performance. The images of eyes have been warped to be roughly of constant position, orientation, and scale. These warped images were then cropped to be of size 20×25 pixels (total 500 pixels). They are represented as 500 dimensional vectors. There is also a set of non-eye images in the training and test sets, which are used to tune and test the detector.
- **run.m**: script for loading the data and calling the functions below. Check this script to understand how the various training and testing functions are called.
- **trainNN.m**: code to train the neural network. **You have to complete this script!**
- **testNN.m**: code to evaluate the neural network. **You have to complete this script!**
- **forwardPropagate.m**: code that must be called by trainNN.m and testNN.m to execute forward propagation. **You have to complete this script!**
- **backPropagate.m**: code that must be called by trainNN.m to execute back propagation. **You have to complete this script!**

- **tryEyeDetector.m**: apply your eye detector on an image (see Figure 1). **You don't need to edit this script.**

What You Need To Do (15 points in total)

[2 points] Complete the **forwardPropagate.m** script to implement forward propagation through fully connected layers with sigmoid functions, as discussed in the class. The script takes as input all the node values from a previous layer in the net, transforms them through sigmoid functions, and outputs the transformed values (check also the comments in this script and lecture notes to understand the input/output per layer).

[5 points] Complete the **backPropagate.m** script to implement backpropagation per each layer in this network. The script takes as input messages backpropagated from an 'upper' layer, and outputs parameter derivatives for the layer specified in the script, as well as messages to be sent to a 'lower' layer in the net. Remember to divide the derivatives by the number of training examples. In addition, use **L2 regularization** on the layer parameters based on the weight decay argument given in the script (don't divide the regularization term with the number of training examples). Note that the script is used for backpropagation for both the output and internal layers of the net. The messages emitted by the output layer should be computed in a different manner than the messages emitted by the internal layers - think how the messages differ for the output layer. There is an argument 'is_output_layer' given in the script that you may use to differentiate between output and internal layers.

[4 points] Complete the **trainNN.m** to train the neural network. Use 3 hidden layers as specified in the default options: 16 nodes in the first hidden layer, 8 nodes in the second one, and 4 in the third one. Call the forward and back propagation functions appropriately. Implement gradient descent updates with momentum for the net parameters. Use the learning rate and momentum based on the default options. Terminate training if the number of gradient descent iterations reaches the maximum number specified in the input options of the script (1000 iterations).

[1 point] Complete the **testNN.m** to evaluate the neural network on test images. The script should call forward propagation and output a binary prediction per test input (eye or non-eye).

[3 points] In the written part of your assignment, report the **classification error in the test dataset** (use the testNN.m script). Show the **ROC curve** illustrating the performance of your detector as its classification threshold is varied. **Try the script tryEyeDetector.m** to use your detector in the test images 'star_trek1.pgm' and 'star_trek2.pgm' (check run.m to see how it is called). **Show the resulting eye detection for these images in your report**, and **mention the threshold you used for classification** for those. Explain **why the threshold might need to be selected different from 0.5**. Note: the tryEyeDetector.m script assumes that you have the image processing toolbox installed - it uses Matlab's Harris corner detector implementation. If you don't have the toolbox, you can download the corners [here](#) (precomputed for 'star_trek1.pgm' and 'star_trek2.pgm') and follow the instructions in tryEyeDetector.m to use them.

Bonus (+2 points in total)





[+0.5 points] Implement batch gradient descent for optimization. Feel free to select the batch size yourself.

[+0.5 point] Implement rectified linear units as activation functions in your network instead of sigmoids.

[+1 point] Train multiple neural networks with different random weights initialization, and average their predictions for classification (this is called 'ensemble averaging')

Submission

Please follow the **Submission** instructions in the [course policy](#) to upload your zip file to Moodle. The zip file should contain all the Matlab code plus **a PDF report including your answers for the written part**. Make sure your program runs in Matlab.

-  [corners.zip](#) 
-  [starter_code.zip](#) 

Submission status

This assignment will accept submissions from **Tuesday, October 9, 2018, 1:30 PM**

Submission status	No attempt
-------------------	------------

Grading status	Not graded
Due date	Tuesday, October 23, 2018, 11:55 PM
Time remaining	262 days 10 hours
Last modified	-
Submission comments	► Comments (0)