Programming Task 2

**Task general description**

In this task you will

* Build a pepper or flower classifier by using a pre-trained network (Alex Net) to extract features, and train an SVM over them.
* Try to Improve the classifier using one of several options: using a better network, extract more features, use data augmentation in training, or other ideas
* Test your Accuracy and report your results

The exercise goals: experience building a classification pipe with CNN components. Get acquainted with some possible methods for pipe improvement.

**Datasets**:

You can choose between peppers classification (Dataset collected in IEM-BGU by Polina Kurster and Yael Edan) and flowers classification (dataset collected in Volkani institute). Download your dataset from the Moodle.

Peppers:

האם יש צורך לבדוק שבאימון יש דאטה מאוזן (חיובי ושלילי?)

האם יש צורך לערבב בין האימון לטסט.?

האם בחישוב הטעות צריך להתייחס לסף גדול מאפס?

The data includes 1343 image rectangles cropped from pepper trees. Among these 661 contain peppers and 682 are negatives – no pepper rectangles. The data is in PeppersData.mat. The original images of the trees are not given due to their size. Use the script WatchPeppers2.m to load and look at the peppers (the image switch every time you mouse press over it). Use the first 1000 images for training, the remaining 343 for test.

This data is part of a set collected by Polina Kurster and Yael Edan. It has not been published yet. Hence **THIS DATA IS CONFIDENTIAL** AND N**OT TO BE DISTRIBUTED OR USED FOR OTHER PURPOSES**.

Flowers:

The data includes 473 cropped images of flowers and non-flowers, with corresponding labels. The original images from which the flowerand non-flower rectangles were cropped from are not given, due to their large size. The data is in the file FlowerData.mat. Use the script WatchFlowers.m to watch the cropped images. Use the first 300 cropped images for training. Test on the remaining 173.

Whatever data you use, you will have to resize the images to 224\*224\*3, which is the input size of the pre-trained CNNs on image net).

**Task description and options:**

The Basic pipe includes:

* Feature extraction by running AlexNet on all dataset images, and extracting the features from the layer–before-last (4096 features). The output is a matrix where is the number of neurons of the before-last global layer, and in the number of examples.
* A linear SVM classifier is applied to the features extracted

In addition to implementing the basic CNN+SVM pipe, you should try at least one way to improve its results. Options are:

* Data Augmentation: Add to the training data (but not to the test data) examples created by applying horizontal flip and/or mild cropping of the original training images. A 3D image can be horizontally flipped by taking I2=I(:,end:-1:1,:). When cropping, use random crops of large portions (>80-90%) of the original image
* Using a more complex network: Instead of Alex net, you may try to use more advanced networks like VGG or residual networks. See <http://www.vlfeat.org/matconvnet/pretrained/> for optional networks.
* Extract more features from the network: Try to add the features from previous CNN layers to the feature taken from the before-last layer. For example, you may concatenate the 4096 features of the layer 2-before last to the 4096 features you were using, and work with 8192 features per example
* Fine tune by re-training AlexNet to the task (note that re-training experiments may take many hours)
* Other ideas you may think of…

**Required results**

You will submit the code and a short report of the results.

Code:

* The code is required to run directly, without modifications, on my machine.
* This machine will have the libraries we use on the matlab path and the dataset in the folder '.\Peppers\' or ‘.\Flowers’ (i.e. it will be a sub-folder in the folder in which the code runs).
* The code is expected to be with similar structure to what we have discussed in the previous assignment, i.e.: it should include stages of preparing experiment parametrization, getting the data, representing it (using CNN this time), train, test, evaluate and report. It is expected to be reasonably documented.
* The code should implement the best version you have found and configured, as long as it runs in less than ¼ hour (so I can check it). If your best configuration is slower, report its results in the report, but keep the code at a version finishing in less than ¼ hour.
* The code should print to the matlab prompt as output the test error result (in a clear sentence), and a figure with Precision-recall curve obtained.

Report: The report should include

* The error rate obtained on the test set, before and after the improvement technique you have tried
  + Report the results of the best pipe configuration you have found for the two cases. If hyper-parameter tuning search was done, report its findings.
* A recall-precision curve:
  + The SVM classify by comparing the score to 0. However, we can change the threshold and comparing to another threshold . For larger than 0, this will classify less examples as ‘object’ and for it will classify more examples as ‘object’.
  + Define
    - Recall: – the fraction of correctly detected objects
    - Precision – the fraction of objects among the algorithm’s positive outputs (i.e. the examples the algorithm claims to be objects)

( is the number of examples the algorithm claims to be objects)

By varying from the lowest obtained score () to the highest one (), we can obtain a recall-precision curve composed of achievable (recall, precision) points. Plot the full recall-precision curve with recall on the x-axis, precision on the y-axis.

* Error images:

The 5 worst errors of type 1 and type 2 (on the test set)

* + Type 1: miss-detection: the algorithm thought it is not an object, but it is
  + Type 2: false alarm: the algorithm thought it is an object, but it is not

For each error, print the error type (1 or 2), error index (1-5) and the classification score (the SVM score) in its title.

**Public code we use:**

* CNN code: We will use MatConvNet, a software package for training and using neural networks.
  + Home: <http://www.vlfeat.org/matconvnet/>

You can either download the code from the home site or install it (requires running a compilation script and assumes you have a C compiler), or download a compiled version from the Moodle.

It seems that there are compilation issues with the newest version (21), so we use version 17. Download link:

<http://www.vlfeat.org/matconvnet/download/matconvnet-1.0-beta17.tar.gz>

* + Pre-trained Networks:

Alex net: <http://www.vlfeat.org/matconvnet/models/imagenet-caffe-alex.mat>

Read in <http://www.vlfeat.org/matconvnet/pretrained/> and See the example in <http://www.vlfeat.org/matconvnet/quick/> of how to use pre-trained models.

* + Documentation: <http://www.vlfeat.org/matconvnet/matconvnet-manual.pdf>
  + Training code example: : <http://www.vlfeat.org/matconvnet/training/> (for the case you wish to re-train)
* SVM code: We will use the SVM from university of East Anglia, as we have done in the previous exercise.