Submission details

- This assignment should be done in pairs (contact the TA if this is a problem).
- The topics of this assignment are filtering, K-Means and Edge Detection.
- The submission date is **29/4/2017**. Late submission will not be graded.
- Coding should be done in Matlab or Python (2 or 3).
- For submission, package up your code as a zip file. Include your written answers as a pdf file named writeup.pdf. Include graphs and images in your writeup, and please label them so we know which figures go with which sub-problem.
- Submit the final zip file through moodle.
- If you have any questions about this assignment, please contact the TA: Alon Shoshan <alonshoshan10@gmail.com>

Task 1: Deep Learning Introduction

Dry section

You should answer in short, 1-3 sentences for each section

1. Working with a convolution network –
   a. Write the output dimensions of every layer for a network with 3 layers:
   - Layer 1: convolution with 64 kernels of size 1X1X3:
   - Layer 2: max pooling of size 2x2:
   - Layer 3: convolution with 32 kernels of size 5X5X64 (no zero padding).
   - Input: RGB image of size 128X128X3:

   b. Explain the calculation of a 2D convolution with a kernel of size 1X1X(?) (a simple example will suffice).

   c. Convolve the following sub-image with a normalized 3×3 filter of your choice (select and design a filter). Write which filter you chose. What is the convolution output? Show the results for two options of stride and padding.

$$\begin{matrix} 4 & 1 & 6 & 1 & 3 \\ 3 & 2 & 7 & 7 & 2 \\ 2 & 5 & 7 & 3 & 7 \\ 1 & 4 & 7 & 1 & 3 \\ 0 & 1 & 6 & 4 & 4 \end{matrix}$$

2. Define three of the following terms:
   a. **Convolution layer**: what are the kernels? what are the parameters? What are the hyper-parameters that the user needs to tune for this layer?
   b. **Pooling Layer**: explain what are average and max pooling. What are the hyper-parameters that the user needs to tune for this layer?
   c. **Fully connected layer**: What is the relation to convolution layer? Can a convolution layer implement a fully connected layer and vice versa?
   d. **Non-linear activation functions**: give 3 examples, add the graphs for each function. Calculate the derivatives for each function and show graph.
   e. **Loss function**: give two examples of loss function. Define each parameter that you use and its size. What is the difference between *regression loss* and *classification loss*?
3. Explain the task of *image classification* and what is the ImageNet challenge?
4. Select on architecture suggested for image classification (e.g., one of VGG19, VGG16, AlexNet) and write the input and output sizes of each layer. If you make assumptions, e.g., of the stride and padding, please explain. We recommend that you search online for details, there are many resources.
5. What is overfitting and how can it be recognized during training? Draw a schematic accuracy vs. iterations graph. Discuss the different cases.
6. How do the learned parameters get updated during training?
7. What is the difference between *train* and *test*? Do they take the same runtime?
8. Bonus (3pt): explain the term batch normalization, write down the formulation.

Wet section

It might be that this is slightly easier to solve in MATLAB, yet 99% of deep learning researchers and developers are working in pytorch\tesorflow thus it is worth the effort.

For this section you will need to use MATLAB Neural Network Toolbox if you are using MATLAB or pytorch\tesorflow if you are using python. You will not need a GPU (basic one should work) for implementing this section. You will probably need to install the tools above first.

*Keep in mind that some tools are harder to learn and get started with than others. This section was designed so it could be run in MATLAB without previous knowledge of the Neural Network Toolbox.

-Examples of MATLAB Neural Network Toolbox: www.mathworks.com/products/neural-network/videos.html
-For installing pytorch: http://pytorch.org/
-For installing tensorflow: www.tensorflow.org/install/

1. Download a pre-trained VGG16:
   For MATLAB you can use this example
   www.mathworks.com/matlabcentral/fileexchange/61733-neural-network-toolbox-model-for-vgg-16-network (try the code. Most likely you will get an error message. Try installing the network from the link you get in the error).
   For pyTorch you use this link: http://pytorch.org/docs/master/torchvision/models.html
   For tensorflow you can use the slim library (has pretrained VGG16):
   https://github.com/tensorflow/models/tree/master/research/slim
2. Load the images in the attached *birds* file and display them. Transform the images so they would fit as input to the network and show the transformed images. Think about size, type, normalization and range. Use the images as input to the network. What are the outputs?
3. Find an image on the web, use it as input, and show the output of the network for it.
4. Use one geometric transformation, one color transformation and one filter on your image (the image from item 3). Show the resulting images. Then use them as input to the network and show the outputs. What has changed with respect to item 3?
5. For 2 filters in the first conv layer of VGG16 show the filters and show their response for the 3 images from item 4.
6. For every image in *dogs* and for every image in *cats* extract the feature vector of layer FC7. Plot all the vectors on the same graph. Explain the result. Can you draw a line (not straight) to separate the vectors?
7. Take one image of a cat and one image of a dog from the internet and show them. Extract their layer FC7 feature vector and find the nearest neighbor [=closest vector] from the feature vectors of item 6. You can select any distance measure, L1 for example, state it in your answer.
8. Take one image of a wolf and one image of a tiger from the internet and repeat item 7 for these images (i.e., find the most similar cat and most similar dog of each image).
9. Build a classifier that separates dogs from cats: choose a layer of the network, collect its outputs for all images. Use these vectors/tensors as the input to an SVM classifier. Use the 20 images of dogs and cats as training set for the SVM and the 2 images from the internet as test set, show your results. You can use any other classifier instead of SVM (you are not asked to explain the classifier details).
10. Take one image of a wolf and one image of a tiger (from item 8) and show your classifier result for them.

Task 2: Edge detection

In this question you will gain some experience with edge detection and with performance measurement of it. You should demonstrate your results on the three provided images.

A. Given an image, an edge detection process should ideally detect the boundaries of all important objects. To see how edge detectors function in practice, we ask you to apply three edge detectors, Sobel, Gaussian-Laplace and Canny, to the gray level images.
How to implement the detectors:
- Canny is fully implemented in both Matlab and Python (under skimage.feature library).
- Sobel filtering is implemented in both Matlab and Python (under skimage.feature, but don't forget global thresholding).

Apply the detectors to the 3 images you select and tune the parameters of each detector manually such that they give the best edge image according to your subjective opinion.

1. Explain in a few sentences the way of operation of each of the three detectors.
2. Explain the threshold parameter and the sigma parameter of the detectors. What happens when you increase/decrease each of the parameters? Which parameter values work best?

B. To measure the performance of an edge-detector, it is common to compare the detection results with those chosen manually by a person. The comparison is between two sets, the set E of pixels detected as edges and the ground truth (GT) set of pixels selected manually. To compare the sets, it is common to use two measures, Precision (P) and Recall (R):

$$P = \frac{|E \cap GT|}{|E|} \quad R = \frac{|E \cap GT|}{|GT|}$$

An edge detector usually depends on a parameter. Usually, the edge detector process calculates some measure of spatial change (e.g. gradient magnitude) and then accepts the pixel as an edge if the measure exceeds a threshold. Note that for very small thresholds almost all pixels are detected, including most GT edges but also many other pixels (small P, large R). Conversely, for large thresholds, many of the pixels detected are true edges but also many of the true edges are missed (large P, small R).

For good performance, both precision and recall are important. Therefore, it is common to express performance by a precision-recall curve, parameterized by some parameter (e.g. the threshold). Another way is to take an average of both. The common average measure is called the F-measure:

$$F = 2\frac{PR}{P+R}.$$

1. Explain in 2 sentences why both Precision and Recall are important.
2. Using the GT pictures provided, plot a precision-recall curve averaged over all images. Plot the curve for each method while taking the threshold as the varying parameter (i.e., one plot, with three curves for every image). Choose the other parameters in any reasonable way (no need to optimize them).