Computer Assisted Image Analysis II
Spring 2018

Excercise 3:

Deep Learning & Motion Tracking

In this exercise you will be given the opportunity to explore how to do tranfer learning on a pre-trained AlexNet and visulaize features from the network. Furthermore, you will have the opportunity to solve a motion tracking problem.

Formality

- You need an account for the computer systems at the Dept. of Information Technology.
- You should work together in groups of two people. It may be advantageous to have someone to discuss issues with.
- You are requested to prepare a written report in pdf format of the exercise answering all questions including explanatory illustrations. Only one report needs to be turned in per group with names of persons mentioned in it.
- The report is handed in via the Student Portal under 'File areas' → 'Lab reports'. Make sure that you belong to a group before starting the lab. You join a group in the Student Portal by going to 'Group divisions' → 'Lab/Project groups'.
- Status of your reports will be found at the Student portal under 'Progress' → 'Progress, mandatory tasks'.
- **Deadline: 28, Feb 2018**

Getting started

In a lab at the Dept. of Information Technology running Windows: Log on to one of the workstations. MatlabR2017a is found under `ITC-Application Explorer(Zenworks)`

As the material for this lab is about 0.5 GB use the following link to download the lab material from Lab 03

1 Transfer learning using a pre-trained Network

Note: This task takes about 24 hrs on the lab PCs so start on time.

In this task we will try to use AlexNet [1] pre-trained on ImageNet dataset [2]. We will start with the `transferLearning.m`. It allows you to choose between two datasets: *MNIST digits (0-9)* and *Birds*. `LOAD_MNIST=1` selects *digits* and `LOAD_MNIST=0` selects *birds* dataset respectively. We choose *digits* dataset for now, *birds* data set can be used for the extra task. The code in `transferLearning.m` is divided into blocks.

`%% BLOCK-n (begin) ******`
`%% BLOCK-n (end)`
where **n** is the block number and `******` is the comment on what the code snippet within the block does.

You will be uncommenting the code within each block according to the instructions. The names of functions underlined are utility functions from NN toolbox in Matlab. Read the documentation from MatlabR2017a for help on them. For deep learning basics in Matlab we suggest that you start here

1.1 Data pre-processing

- STEP 1: Uncomment the code in the following blocks

  # `%% BLOCK-1 (begin) Load pre-trained Alex-net`
    - This loads `net` variable into the workspace. `net` is a pre-trained AlexNet on ILSVRC dataset.

  # `%% BLOCK-2 (begin) Generate data-structure and labels for images`
    - `imageDatastore` creates labels for images based on the parent folder name. It returns a data-structure with file names and a read function to load images along with the labels.
    - `splitEachLabel` performs a training/validation split in the dataset.
    - Rest of the code is to visualize the images from the dataset.

  # `%% BLOCK-3 (begin) Network surgery`
    - Here we truncate the last 3 layers in the network that correspond to classification in ILSVRC dataset and retain rest of the network.
    - We add new layers to classify our current dataset as per the number of classes we have.
    - We set the mini-batch size and iteration lengths.

  # `%% BLOCK-4 (begin) Network training`
    - `trainingOptions` sets the options required for training the network. Set the `'CheckpointPath'` to point to the `log` directory provided. This is where intermediate trained models are saved. It is important to note that we do not need too many epochs to converge so `'MaxEpochs'` is set to 4 and we need a slow learning rate for the new layers so `'InitialLearnRate'` is to 10^{-4}
    - We read the input images into a 4D data array.
    - `trainNetwork` is used to train the network.
    - `classify` is used for prediction.

1. ***There are two problems to fix in*** `%% Prepare 4D data array` ***for the training to start. One of them is related to the network architecture, and the other has to do with practical constraint on system memory respectively. Comment on the code fixes.*** Hint: Use the `augment_data` function with proper arguments. Pass `type='basic'` and `max_sz=300` for a reasonable training dataset size.

2. ***Attach the plot for training accuracy. It should look like Fig. 1***

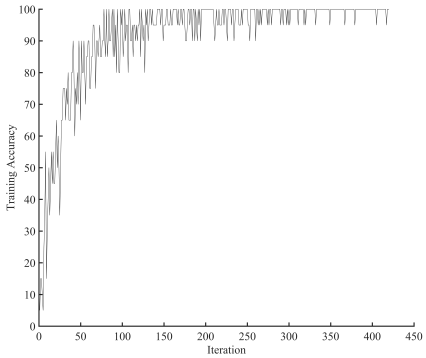

Figure 1: Training accuracy for various iterations.

1.2 Visualizing the features

For this task you can either

- - Save `net` and `netTransfer` variables in your workspace using `save('my_netTransfer.mat','net','netTransfer');`
  - Modify `visualizeFeatures.m` to load the `my_netTransfer.mat` and run it.
- Or run `visualizeFeatures.m` directly which visualizes features from pre-trained network on digits.

3. ***Attach the feature visualizations by exporting the images. They should look like Figs 2 - 7. What can you see in the visualizations? Comment on the nature of the features learned at*** `conv1, fc6, fc7` ***layers.***

4. `conv1` ***features look almost identical between AlexNet and the modified AlexNet for transfer learning on*** **MNIST digits** ***and feature representations in layers*** `fc6, fc7` ***look different, why?***

2 Motion Tracking

Motion analysis is similar to image registration since estimation of velocity could be compared to estimation of disparity. However, in motion analysis we usually have:

- A temporally ordered sequence of images
- These images have the same modality
- Often, two consecutive images are similar

For this assignment you will be working with an image sequence depicting prawns moving around in doughnut shaped container. The image sequence is taken from the dataset of an ongoing research project where the movement of the prawns is studied.

Change directory to `Motion`.

The image sequence you are going to analyse, named `source_sequence.avi`, consists of 581 frames. The first 250 frames could be considered as "easier" whereas the remaining frames contain a number of difficulties. To help you a little a mask (named `bwmask.png`) has been supplied that can be used to remove the central part of the frames if needed. An example result has also been supplied, named `example.avi`.

1. ***Create an algorithm that is able to track the prawns for the first 250 frames. For the report specify how you solved the problem. Include the relevant parts of the code in the report and a video capturing the final tracking.*** Hint: ***Take a look at some of the suggested functions below.***

2. ***Now run your algorithm on the final frames of the sequence. Does it fail in any way? If it fails what is the problem? Write a short summary of the problems that arise in the second part of the sequence and suggest some possible solutions (you do not need to implement them).***

2.1 Some help

Some useful functions and code snippets (see the help documentation for each function to see what it does):

- `VideoReader`
- `imextendedmin`
- `regionprops`
- Code snippet for saving an image sequence similar to `example.avi`:

```
aviobj = VideoWriter('Test.avi');
aviobj.Quality = 100;
aviobj.FrameRate = fps;
open(aviobj);
%begin loop
fig = figure(10);
imh = imshow(frame);
hold on;
line(x,y,'marker','o','markersize',10,'LineWidth',2);
F = getframe(fig);
writeVideo(aviobj,F);
%end loop
aviobj = close(aviobj);
```

3 Extra Task

Try to repeat Task 1 (Transfer learning using a pre-trained network), now using *birds* data set. You will have to use data augmentation by passing `type='rotation'` in the `augment_data.m`. Check the classification accuracy on this task with and without data augmentation. Observe and comment how the features vary in each case.

References

[1] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[2] J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, and L. Fei-Fei. ILSVRC-2012 , 2012. URL http://www.image-net.org/challenges/LSVRC/2012/ .

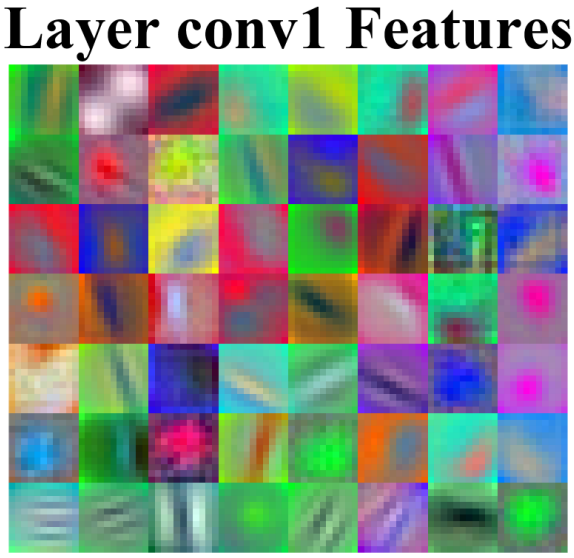

Figure 2: Pre-trained AlexNet conv1 features.

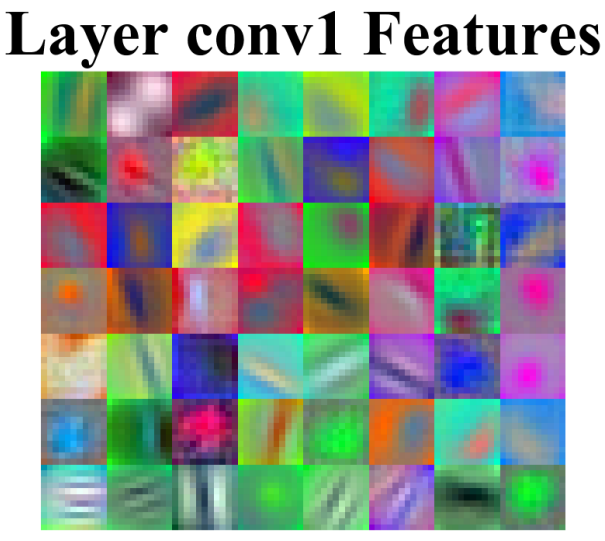

Figure 3: Transfer learning AlexNet conv1 features.

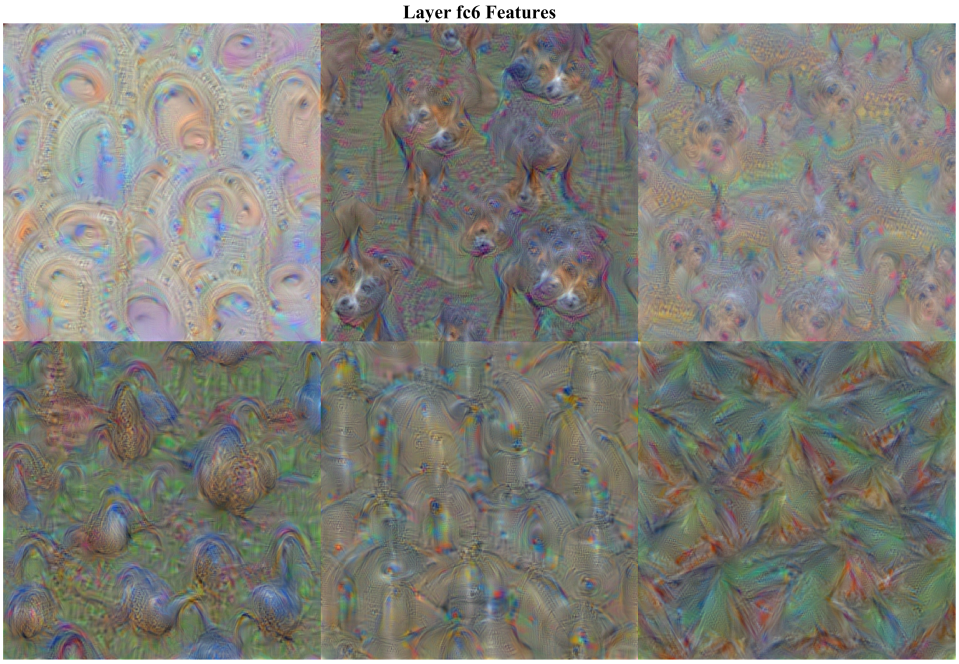

Figure 4: Pre-trained AlexNet fc6 features.

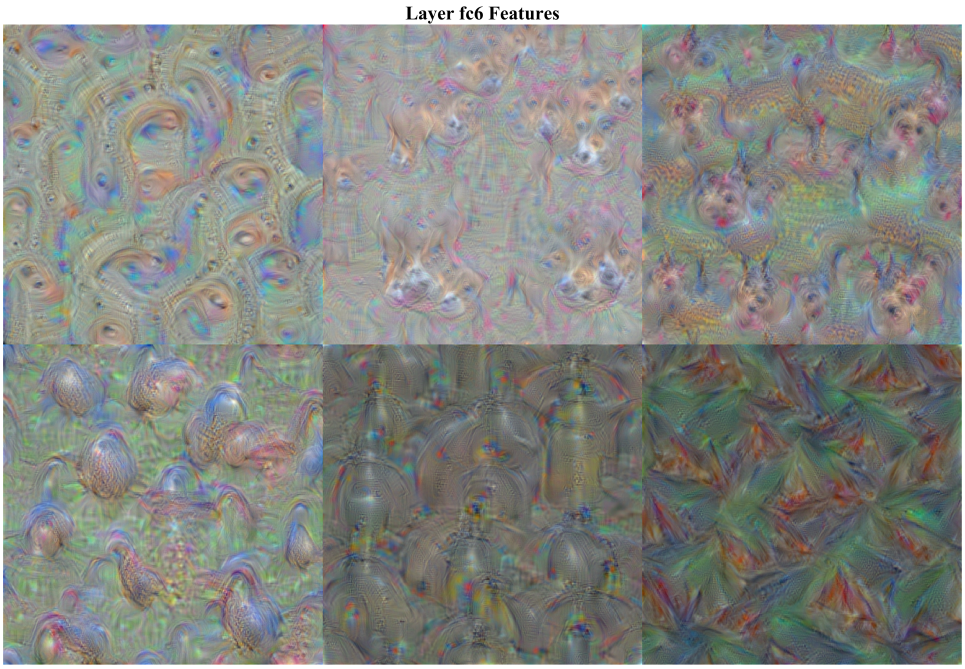

Figure 5: Transfer learning AlexNet fc6 features.

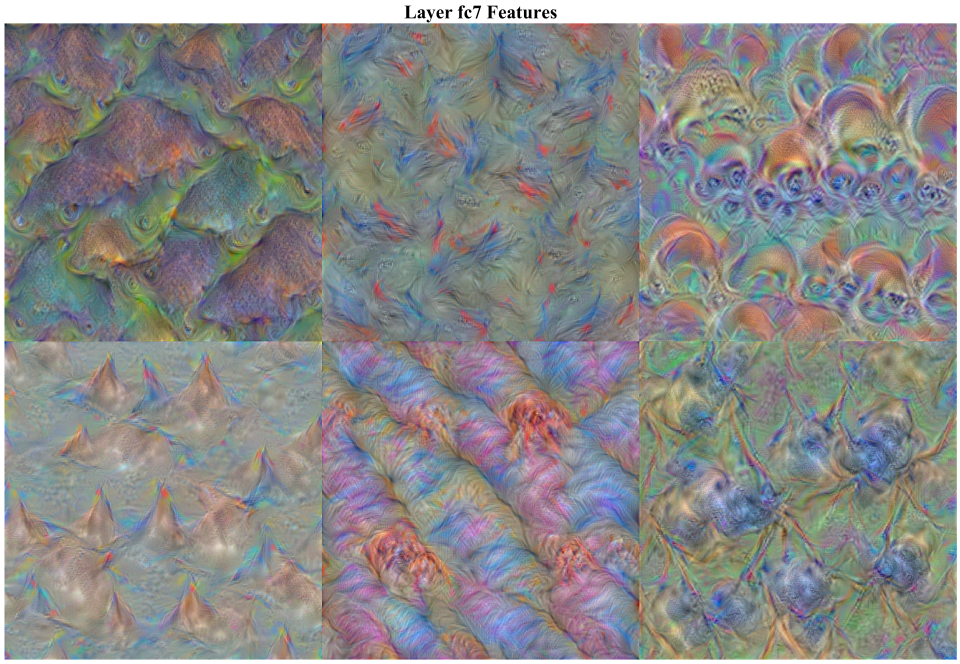

Figure 6: Pre-trained AlexNet fc7 features.

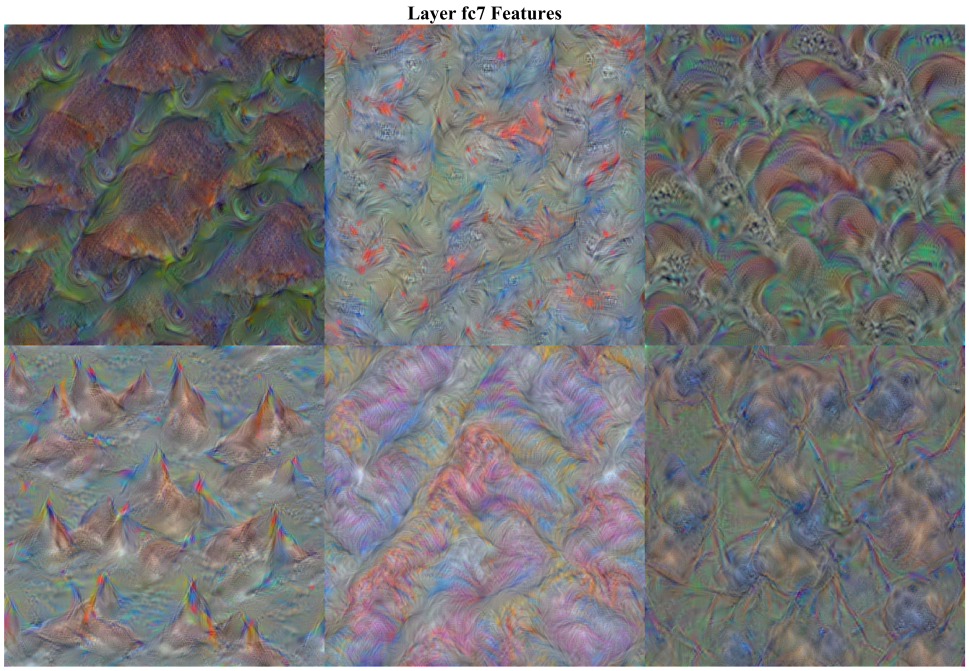

Figure 7: Transfer learning AlexNet fc7 features.