Exercise Sheet 5: What does a neural network see?

Due on 31.05.2024, 10:00

Ulrich Prestel (U.Prestel@lmu.de)

Important Notes:

1. **Questions**: Please use the Moodle platform and post your questions to the forum. They will be answered by us or your fellow students.
2. **Submission**: Put your code and potentially other materials inside a single ZIP file. If you use Jupyter notebooks, please always create a PDF file and include it in your ZIP file. The final submission should therefore be a **single zip file** with a **PDF of your code** and the **original code** inside. The ZIP file should contain your surname and your matriculation number *(Surname-MatriculationNumber.zip)*. **Submissions that fail to follow the naming convention will not be graded!**
3. **Deadline**: The due date for this exercise is the 31st of May.

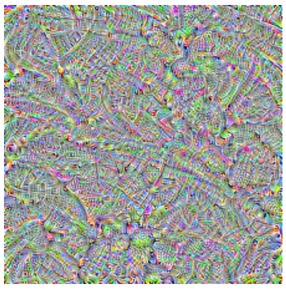
Task 1: Filter Visualization (10P)

Let us have a look inside a convolutional neural network to identify what different filters are responsible for.

1. *VGG16.* As the first task you need to load the pretrained VGG16 network. Please familiarize yourself with the network architecture and print short model description.

2. *Optimize the input image.* In the lecture you have learned that the training of neural networks is performed by means of gradient descent on the network weights with the objective to minimize the loss function on the input batch of images. This process can be actually turned around: we can change the input image values in order to maximize output activation. Familiarize yourself with the code in file `filters_visualization.py`. Print the shape, mean, minimum and maximum of the intermediate network-activations.

3. *Visualize a random filter.* Now we can actually apply the optimization process to the input image pixels to maximize activations of the selected filter in the selected layer. Run this process multiple times for different filters `filter_nmbr` and visualize activations. Your results should look similar to Fig. 3.



4. *Visualize filters at different layers.* Now repeat the process for for different filters `filter_nmbr` **and** different layers `layer_nmbr`. Plot those activations. What do you observe going from earlier layers to later layers?

5. *Hyperparameter tuning.* The process of visualization is quite sensitive to the parameters. Some of the crucial parameters to tune are number of optimization steps `num_optim_steps`, learning rate `lr` and type of an optimizer `optimizer`. Vary those and describe what you see. Another important parameter is the input image `rand_img` initialization. Play with low and high values used for initialization and describe what you see. In particular try out an image filled with the value 128. What do you observe?

6. *Pretrained weights.* The filters' visualizations you observe are a result of a long and computationally expensive training of the VGG16 network on the millions of images. You can try to visualize the filters without having pretrained weights: change pretrained to `False`. What do you observe?

Task 2: Deep Dream (10P)

1. *Most activated filters for every image.* Now we can establish a connection between the actual image content and what our neural network sees in those images. Download 5 random photographs and fix a particular layer `layer_nmbr` of VGG16. You can directly replace the `rand_img` random noise with an actual image and analyze the `conv_output` tensor without indexing the filter with `filter_nmbr`. Now simply select the 10 most activated filters for each content image. visualize these filters by using the technique designed in the previous exercise. What do you observe?

2. *Total variation loss.* One of the problems with the visualization of the filters is that the optimization image becomes too noisy. We can fight that by directly adding the total variation loss for image I:

$$TV(I) = \sum_{i,j} \sqrt{|I_{i+1,j} - I_{i,j}|^2 + |I_{i,j+1} - I_{i,j}|^2} \tag{1}$$

   This loss computes the difference between neighbouring pixels. If we add this loss we can enforce additional image smoothness. Visualize 5 filters optimized with and without the total variation loss. You may need to adjust the weight of the total variation loss to make it visible.

3. *Deep dream.* Till now we were maximizing a single filter starting from a chosen input-image. We can however do two important modifications: 1) change the input image to a real photograph and 2) maximize activations of all the filters. For numerical stability you better use the L_2 norm of the activations tensor and try to maximize this value during optimization. Moreover, you need to add total variation loss and play with hyperparameters a bit to obtain the best visual quality. The best performance is usually achieved if you maximize activations not of one but of multiple layers. Try this out as well and report results.