Exercise Sheet 8

Convolutional Neural Networks

Deadline: 17.01.2017, 23:56

Convolution

Exercise 8.1 *(1+1+2+1+2=7 points)*

Consider the following 1-D signal

$$f = \boxed{3 \mid 1 \mid 8 \mid 6 \mid 3 \mid 9 \mid 5 \mid 1}.$$

Furthermore consider the kernel

$$k = \boxed{\tfrac{1}{2} \mid \tfrac{1}{2}}.$$

a) Compute the discrete convolution of the signal f with kernel k, i.e $s_1 = f * k$.

b) Compute the discrete convolution of s_1 and the kernel k, i.e. $s_2 = s_1 * k$.

c) Provide a plot of f, s_1 and s_2. Briefly describe the effects of convolving f with the kernel k. What would be the outcome of convolving f with k for n times when $n \to \infty$?

d) Compute a new kernel k' as $k' = k * k$. Now compute $s_3 = f * k'$.

e) Compare s_2 and s_3. What do you notice? Which mathematical concept seems to be fulfilled here by convolution? Prove that this concept is fulfilled by convolution for arbitrary signals.

Convolution and Convolutional Neural Networks

In this section we want to get a feeling for the similarities between a standard 2-D convolution as it is used e.g. for image processing and its corresponding implementation as a Convolutional Neural Network.

Exercise 8.2 *(4+1+1+1=7 points)*

The goal of this exercise is to implement a standard 2-D convolution as defined in lecture 8, slide 5. Therefore download the password protected `zip`-archive `ex08.zip` from the course website. The password is `psr_ws_1617`. The archive contains some images used later. To import images as a `numpy`-matrix in `python` you can use the `misc` module from `scipy` (`from scipy import misc`). With `im = misc.imread(<path>)` you obtain the matrix `im` as a matrix containing the color values of the image `<path>`. You can display a gray value image `im` using `matplotlib` (`import matplotlib.pyplot as plt`) with the function call `plt.imshow(im, cmap=plt.cm.gray)`.

a) Supplement the python script `convolution.py` which performs a 2-D convolution with a given 2-D signal $f(x, y)$ (e.g. a grey value image) and a kernel $k(x, y)$. You can assume that the kernels will have a shape like

$$k(x,y) = \begin{array}{|c|c|c|c|c|} \hline \dots & \dots & \dots & \dots & \dots \\ \hline \dots & (-1,-1) & (0,-1) & (1,-1) & \dots \\ \hline \dots & (-1,0) & (0,0) & (1,0) & \dots \\ \hline \dots & (-1,1) & (0,1) & (1,1) & \dots \\ \hline \dots & \dots & \dots & \dots & \dots \\ \hline \end{array},$$

i.e. they their center weight corresponds to the index $(0, 0)$.

As indicated on slide 6 the resulting image will have smaller dimensions than the original one, depending on the kernel size. To overcome this issue assume that the original image is padded with as many zeros as needed to preserve the original dimensions.

As the result of the convolution may not be in the range $[0; 255]$ anymore apply the function `min-max-rescale()` to the result, before displaying or saving it.

b) Apply the kernel

$$k_1(x,y) = \frac{1}{9} \cdot \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

to the image `clock_noise.png`. Describe the effect of the kernel and an application where such a kernel can be used for.

c) Apply the kernel

$$k_2(x,y) = \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 1 & -2 & 1 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

to the image `clock.png`. As before, describe the effect of this kernel and possible applications. Which meaning do different gray values have?

d) In part a) we padded the image with zeros in order to obtain an image with the same size as the original one. Why is this not always optimal? Give an example where this leads to unwanted results.

Hint: Think of the kernel k_2 from c). Which mathematical concept does this kernel implement. Which effect does the zero padding have for border pixels? What would be a more suitable padding in this case?

Exercise 8.3 *(3 points)*

The goal of this exercise is to build a Convolutional Neural Network using `Tensorflow`.

As our dataset we again want to use the `MNIST` dataset. The images have a size of 28×28. As they come as a vector in the training data you have to reshape them first to a 28×28 matrix.

The weights you are using should be initialized randomly. The biases should be initialized with a value of 0.1.

Now build your network as follows:

- The first hidden layer applies convolution (`tf.nn.conv2d`) with a 5×5 mask to the input nodes. For each of the patches you should produce 32 outputs. In order to retain the size of the layer perform zero padding. As an activation function for the output use the `RELU` function.

  In order to reduce the number of resulting nodes perform pooling by averaging with a 2×2 mask afterwards.

- The second hidden layer should be a fully connected layer, which takes the outputs of the previous layer as inputs. As an activation function for the output again use the `RELU` function. Try different sizes for this layer and comment on the effect to the accuracy.

- The output layer is again a fully connected layer and should yield a vector $\mathbf{x}$ of length 10. Each entry x_i should contain the probability of the given sample belonging to the number $i \in \{0, ..., 9\}$.

As a cost function use the cross entropy (`softmax_cross_entropy_with_logits(...)`) which you should normalize (`reduce_mean(...)`). As an optimization algorithm you can use the `AdamOptimizer` with a learning rate of 10^{-4}.

Train the model on the training data (You do not have to consider the full training set if it takes too long). Provide a plot of the evolving prediction accuracy.

Then evaluate the model on the test set. Report the accuracy the model achieved.

Gabor Filters

Exercise 8.4 *(3 points)*

In this exercise you should make yourself familiar with the Gabor filter.

The image `hide.png` shows a person which is painted such that it is hard to recognize in front of the background. Your goal is to indicate the presence of the person using Gabor filters.

Therefore proceed as follows:

- Implement a python script `gabor.py` which computes the convolution of an image and a Gabor filter. You are allowed to use existing implementations for that. I.e. you can modify this http://scikit-image.org/docs/dev/auto_examples/plot_gabor.html script for your purposes.

- In order to outline the person you might have to compute several filtered versions of the image and average them.

- Report the different parameters you used and provide a plot of the corresponding filters. Also briefly state why you have chosen these parameters.

- Make sure to also include the result in your report.

Submission instructions

The following instructions are mandatory. If you are not following them, tutors can decide to not correct your exercise.

Submission architecture

You have to generate a **single ZIP file** respecting the following architecture:

```
tutorial8_<matriculation1>_<matriculation2>_<matriculation3>
|
+--- source
|       |
|       +------- file 1
|       +------- file 2
|       +------- ...
+--- rapport.pdf
+--- README.txt
```

where

- **source** contains the source code of your project,
- **rapport.pdf** is the report where you present your solution with **the explanations** and the plots,
- **README** which contains group member informations (name, matriculation numbers and emails) and a **clear** explanation about how to compile and run your source code

The ZIP filename has to be :

```
tutorial8_<matriculation1>_<matriculation2>_<matriculation3>.zip
```

Some hints

We advice you to follow the following guidelines in order to avoid problems :

- Avoid building complex systems. The exercises are simple enough.
- Do not include any executables in your submission, as this will cause the e-mail server to reject it.

Grading

Send your assignment to the tutor who is responsible of your group:

- Merlin Köhler s9mnkoeh@stud.uni-saarland.de
- Yelluru Gopal Goutam goutamyg@lsv.uni-saarland.de
- Ahmad Taie ataie@lsv.uni-saarland.de

If you are assigned to different tutorials send your assignment to the tutor to whom most of you are assigned to.
The email subject should start with `[PSR TUTORIAL 8]`