Fred Hamprecht, Alberto Bailoni
cvflecture.s20@gmail.com

General Regulations (updated)

- No individual submission! Hand in your solutions in groups of two or three people (preferred).
- Only include a single jupyter notebook per submission. Not one per person or separate ones per exercise.
- All outputs, in particular images, should be visible in the notebook. We should not have to run your notebook or uncomment lines / change some code to see your results!
- When asked, remember to comment / interpret your results.
- Every result or image you provide should be at least pointed to in the notebook (by file / directory name).
- Include all files/code that include your work in the zip file. Otherwise, it will receive no marks.
- Submit all your files in a single `.zip` archive (not `.rar` or `.tar`) and upload it using the Physics platform https://uebungen.physik.uni-heidelberg.de/h/1176. **Only one person in the group should upload the zip archive**: in the upload interface you will see the option to indicate your exercise-partners.
- **Starting from this exercise sheet, a violation of any of these regulations can lead to loss of points.**

Working with Google Colaboratory

In this assignment, we will start working with PyTorch, a very common deep learning library for python. Next week we will train a CNN model using a GPU. You can rely either on the Google Colaboratory infrastructure (recommended) or on your local GPU if you possess one. Google Colaboratory works similar to the jupyter notebooks we have been relying on throughout the exercises, just that it runs on Google servers and it has PyTorch already installed. Per default this only uses CPUs, but if you go to *Edit→Notebook Settings*, you can switch the "Hardware Accelerator" from *None* to *GPU*.

In the jupyter notebook `ex03.ipynb`, you find more information on how to set up the Google Colab environment.

1 Parameters and Receptive Field of a CNN (4pt)

Consider the simple classification model shown in Fig. 1, which consists of a sequence of a few convolutional, MaxPool and fully connected layers.

(a) Number of parameters (2pt) How many parameters does the model have? Explain your result. In the lecture we did not talk about *bias parameters* of convolutional layers, so you can ignore them when counting the number of parameters.

(b) Receptive field (2pt) Now consider the output of the convolutional part of the model, which is indicated with $(*)$ in Fig. 1. What is the dimension of this output? What is the (theoretical) receptive field of the model before applying the last two fully connected layers? Have a look at this article[1] for an introduction to the concept of receptive field.

[1] https://medium.com/@abhigoku10/topic-dl03-receptive-field-in-cnn-and-the-math-behind-it-e17565212a20

2 Train a CNN for Semantic Segmentation (Part 1) (8pt)

In this exercise, we will work again with images of yeast cells (see Fig. 2 for an example) and train a CNN to perform a foreground-background segmentation using PyTorch[2], a very popular deep learning library for python. In this first part of the exercise we will start preparing the Data-Loader, add some data augmentation and implement the metrics used to evaluate the model. In the next assignment, we will train the model.

(a) Loading the data (2pt) When you work with large data, it is very common to store them on disk in the form of HDF5 files[3]. `h5py` is a python module that can efficiently create `.hdf5` files from `numpy` arrays (and vice versa). In the jupyter notebook `ex03.ipynb`, you find a script to download the YeastCells dataset, which includes 18 `.png` images with associated ground truth labels like the one shown in Fig. 2. Following the instructions in the notebook, create a single `.hdf5` file containing all the data.

(b) Normalization (1pt) It is often useful to normalize your input before feeding it to the neural network. For an input vector, you would normalize each feature to have zero mean and a standard deviation of one over all the data. Create a new `.hdf5` file containing the normalized raw input as described in the jupyter notebook.

(c) Data augmentation (2pt) In the jupyter notebook, we provide an example instance of the `BasicTransform2D` class that downscales an image by a given factor. Implement further two classes derived from `BasicTransform2D` that should do the following: i) reflect an image along x and/or along y with probability 0.5; ii) randomly rotate an image by a multiple of 90 degrees.

(d) Semantic Segmentation Metrics (3pt) Implement the following two metrics that will be used to evaluate the performance of our semantic segmentation model: i) *accuracy*, i.e. the number of correctly classified pixels divided by the total number of pixels; ii) *intersection over union*[4] averaged across all classes (for this task, there are only two classes: background and foreground). Which one of the two metrics do you expect to be more informative? Why?

(e) Train a Semantic Segmentation Model – To be continued in the next assignment.

[2] https://pytorch.org/
[3] More information about HDF5 files: https://portal.hdfgroup.org/display/HDF5
[4] Definition of Intersection over Union or Jaccard index https://en.wikipedia.org/wiki/Jaccard_index

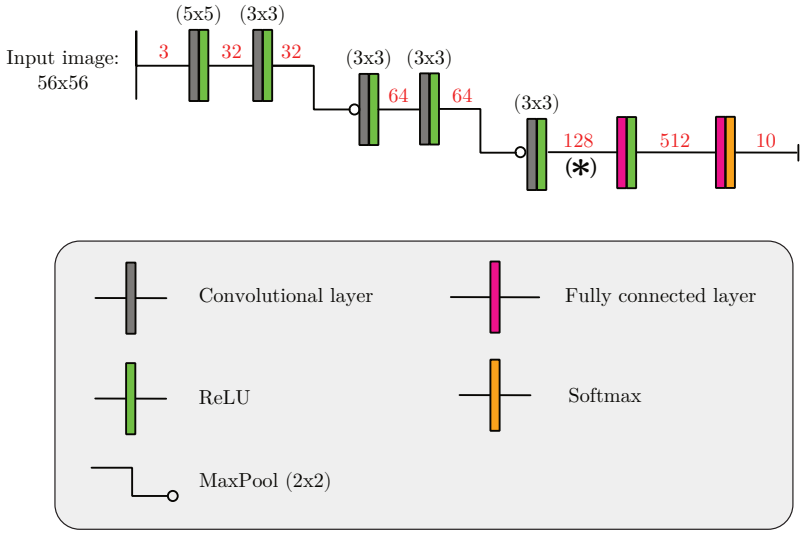
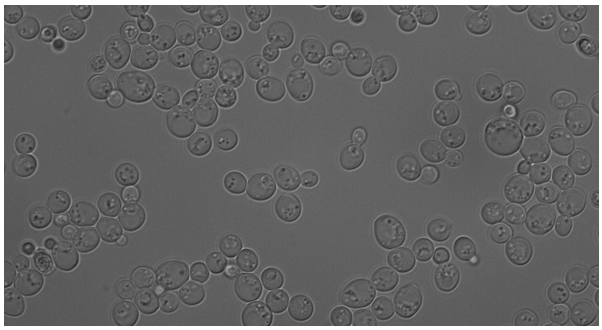
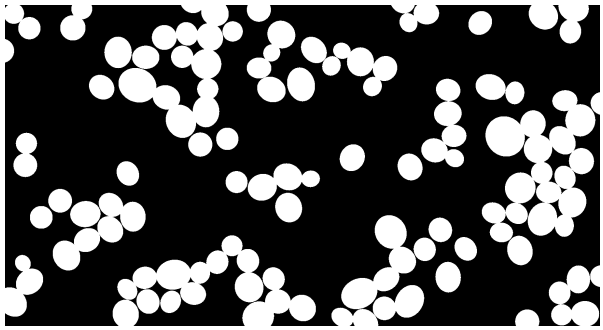


Figure 1: Example of very simple architecture for image classification. The input is an RGB image (three channels) with size 56×56. The numbers of channels are written in red: the number of output classes is 10. For each convolution layer, the dimension of the 2D filter is specified. In the architecture we always use "same" 2D convolutions, so that the input is padded with zeros and the convolution output has the same spatial dimension.



(a) Raw image



(b) Ground truth foreground-background segmentation

Figure 2: Image of yeast cells. Some of the cells in the image are dead and should not be detected, for example the two darker ones in the top left corner. This yeast-cells dataset was kindly made available by Joel Goodman, *UT Southwestern*, Dallas, Texas.