# Machine Learning for Agricultural Applications

## Assignment 7

Prof. Dr. Niels Landwehr
Dr. Julian Adolphs

Summer Term 2020

Release: 18.06.2020
Discussion: 25.06.2020

### Task 1 – Segmentation [10 points]

In this exercise you will work with the Oxford-IIIT pet dataset.[1] The Oxford-IIIT pet dataset is a 37 category pet image dataset with roughly 200 images for each class. The images have large variations in scale, pose and lighting. All images have an associated ground truth annotation of breed. Here we do not want to classify the 37 cat and dog categories, we want to train and predict the segmentation. For that we need training data that are annotated with segmentations masks, which is the case for the cats and dogs of this dataset. The learning task is to predict pixelwise if a pixel belongs to cat/dog, background or margin (Figure 1).

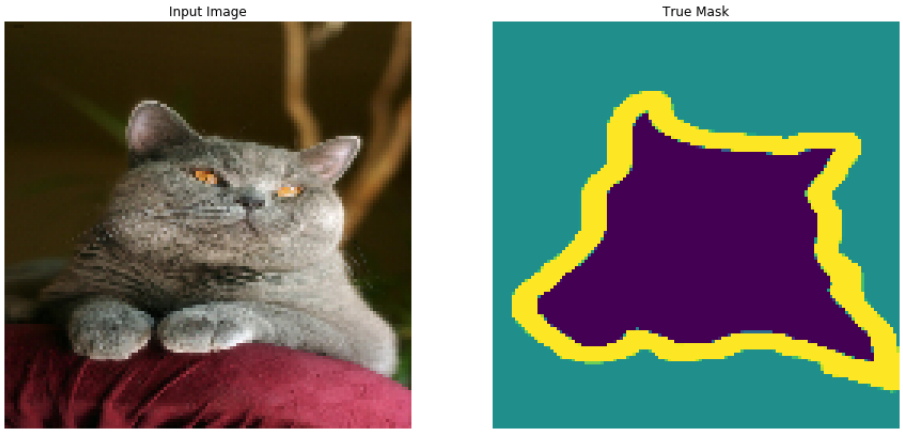


Figure 1: One samples of the Oxford-IIIT pet dataset.

It is recommended to implement this exercise in Colab or a similar cloud computing service, so you have access to a powerful GPU for training the model. The Oxford-IIIT pet dataset is available from github. The first task is to follow the instructions on `https://www.tensorflow.org/tutorials/images/segmentation` and run the program.

[1] IEEE Conference on Computer Vision and Pattern Recognition, *Cats and Dogs*, O.M. Parkhi, A. Vedaldi, A. Zisserman, and C.V. Jawahar, 2012.

**Task 2 – Construct a mini U-Net with Keras** [25 points]

Up to now, we always worked with the Keras Sequential API, that groups a linear stack of layers into a model. The U-Net architecture from the lecture[2] contains shortcuts, which can be constructed with the `concatenate`-command of the Keras Functional API (an easy to understand tutorial is for example:
`https://machinelearningmastery.com/keras-functional-api-deep-learning/`). Use `input_layer = Input(shape = (128, 128, 3))` as input of your model (the images from the dataset have different resolutions, so we use here a minimum resolution, that fits for all). Construct a mini-version of the U-Net explained in the lecture. Start with the resolution of (128, 128) in the first level. If you use `padding='same'`, the image size in the second level is (64, 64) and in the third level (32, 32). Compared to the Ronneberger U-net, you should construct only the three lower levels of Figure 1 of the Ronneberger paper (instead of 5 levels). The use of `padding='same'` makes everything a bit more convenient. The output segmentation map in our case has again the resolution (128, 128) and 3 output channels. Use `model.summary()` and `plot_model(model, to_file='model.png')` to check your mini U-Net.

**Task 3 – Use mini U-Net for Segmentation** [15 points]

Now replace the model that was used in the solution of
`https://www.tensorflow.org/tutorials/images/segmentation` in Task 1 by your mini U-Net construction and run the program again.

Is your mini U-Net performing well on the segmentation task? Compare with Task 1.

Hint: Start with drawing a scheme of your mini U-Net!

[2] U-Net: Convolutional Networks for Biomedical Image Segmentation. O. Ronneberger, P. Fischer, and T. Brox. 2015, arXiv:1505.04597