# Outperforming the data

How to use simple Python visualizations and Keras to quickly learn strong labels on a small, unlabeled dataset

Recently I have been trying to solve a problem for a friend whose getting her PhD in ecology. She was wondering if I could compute the area covered by plants within a box in a set of images. The 8 images were taken from different angles and in different light settings, there was no previous mask function to learn from, and the box was in different locations for each image. I started the way most data scientists would start with a problem like this: I made exploratory plots. Based on some simple visualizations, I determined which parameters would work for a coarse (weak) segmentation mask. These coarse masks where then fed to a Network, which produced, stronger masks, which were again used to update the network, resulting in a strong segmentation of the original images.

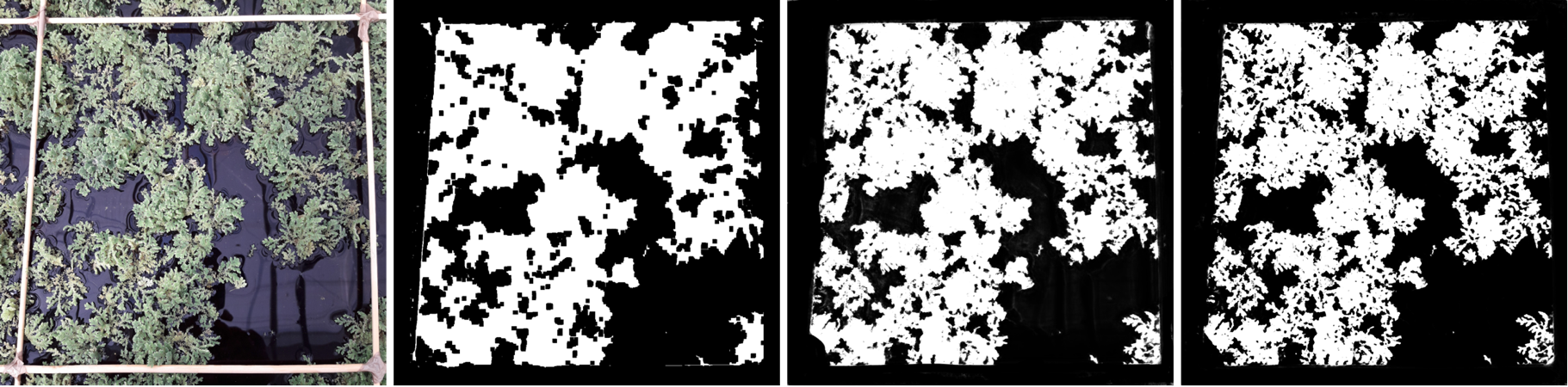
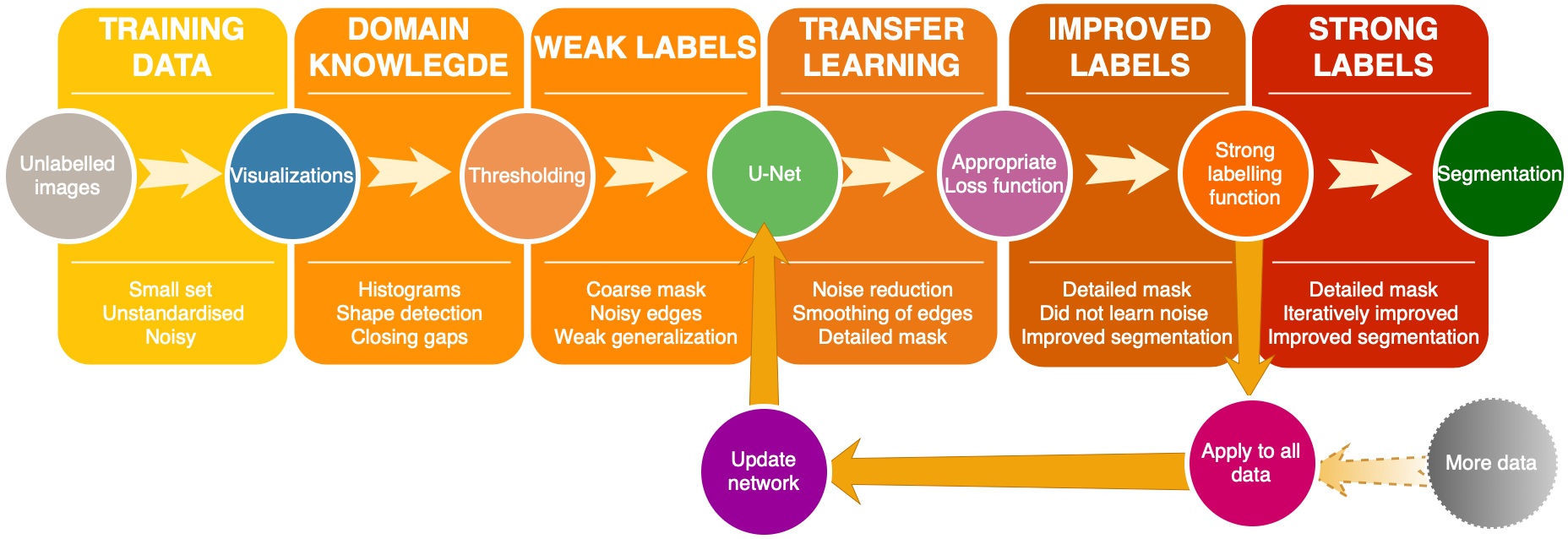
In this blog I will show how I tackled the problem of image segmentation for the tracking of (floating) plant cover in quadrants. Designing a solution for this problem came with a noticeable constraint: a lack of any training data in sufficient quantities. We will find that for certain problems, a lack of data does not have to hinder the creation of a high-performance model, by applying domain knowledge and the careful selection of a loss function.

Figure 0‑1: From raw data, to weak labelling, to stronger labelling and iterative improvement

At the end of this blogpost you will know:

* How we can quickly use visualizations to create a small, weakly labelled dataset;
* How a Keras implementation of U-net can be used for noise reduction and results in stronger labels;
* How careful selection of loss functions allows our network to use limited data to learn to perform the task beyond the original scope of the data;
* Why an intuitive understanding of loss functions and data metrics helps you adapt this learning pipeline to your own dataset and learning task.

# Segmentation for plant phenotyping

Plant phenotyping is the identification of effects on plant structure and function, resulting from genotypic differences and the environmental conditions a plant has been exposed to. Knowledge of plant phenotypes is a key ingredient in the evaluation of, for instance, biomass productivity [1].

While collection of phenotypic traits was previously manual, image-based methods are now increasingly utilized in non-invasive plant phenotyping and the resulting images need to be analyzed in a high-throughput, robust, and accurate manner [2]. There is a need for data at the plant feature level to be able to understand the effect of genomic variants on phenotypes that are needed to decipher the causes of plant health, crop yields, disease and evolutionary fitness. The practical importance of high-throughput automated phenotyping is widely recognized:

*According to the Food and Agriculture Organization of the United Nations, large-scale experiments in plant phenotyping are a key factor in meeting the agricultural needs of the future to feed the world and provide bio- mass for energy, while using less water, land, and fertilizer under a constantly evolving environment due to climate change.*

-Image Analysis: The New Bottleneck in Plant Phenotyping (Minervini et. al.) [3]

The acquisition of high-dimensional phenotypic data on an organism-wide scale differs from the usual tasks addressed by the computer vision community [3]. A quick and cheap way to acquire data is taking two-dimensional images from photography and tradition microscopy, but here limited specimen handling time is often the restricting factor and results in small, noisy datasets.

# Exploratory data analysis

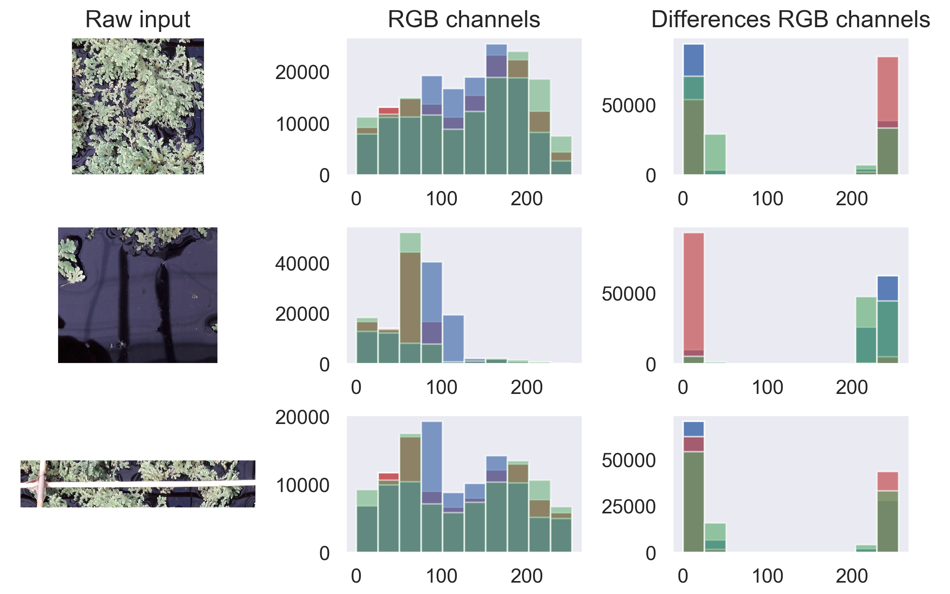
We start with a small dataset that we want to learn the segmentation task for. We have 8 pictures of *Azolla filiculoides* (water fern), taken from different heights and different angles, that have not been labelled yet.

To any person, this looks like a simple task. You separate the plants from the background. Most people, given a GUI that helps them select connecting areas, would perform pretty decently in this segmentation problem. We would like to find a way to 1) minimize the need for manual segment selection, and 2) generalize this manual segment selection to a larger set.



Figure 0‑1: Raw image 1 and image 2 of plants that we want to learn the masking function for

# Preprocessing with explicit domain knowledge

We can generically recognize the plant, box edge, and water covered areas by their color. By visualizing the color channel histograms of certain areas of interest, we can determine the cutoff values for these color bands, if we want to visualize the plant cover only.

The green area has a lot of green values over 100. The dark water area has many low values with small differences. The area containing part of a twig has very few differences between color channels over the whole range of values.

Let’s choose the following rules for preprocessing:

* The green value at a single pixel should be over 100, otherwise remove it;
* The differences between color channels at a pixel should be below 30, otherwise remove it;
* Areas where all values are over 230 are too light and are removed as well.

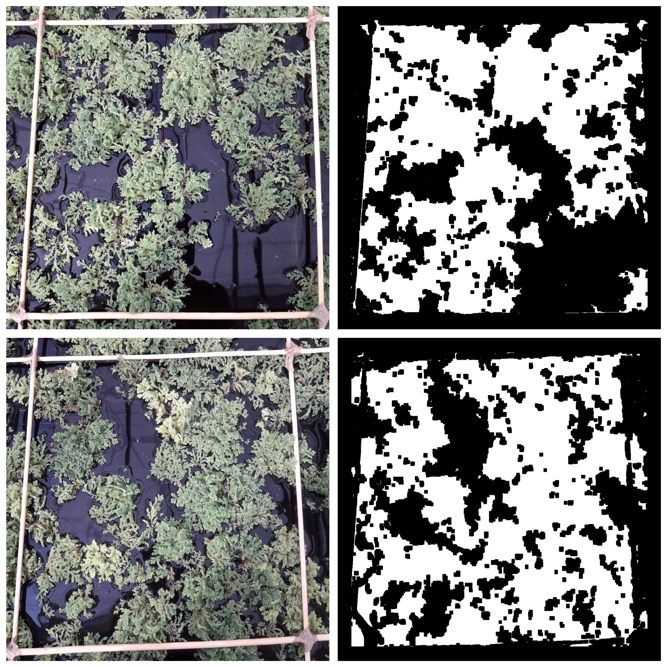


This results in a coarse labelling of the plant cover areas. While most of the plant cover is cleanly separated from the water, a lot of lighter edges are lost.

# Applying implicit knowledge to weak labelling

We use a few python image processing tricks to generalize these rules to other images. First, we want to make sure that we don’t lose too much information around the edges. We do this by applying the morphological closing transformation to the masked area, which will ‘close’ the gaps between pixels that are the same color. Second, we determine where the area of interest within the photographed square is located. Simply performing a coarse area labelling from the scikit image processing library gives us the correct section of the image to use for this learning task.

We accept that this labelling is imperfect and hope that by some data augmentation, we can learn to ‘denoise’ the labels and obtain the correct mask.



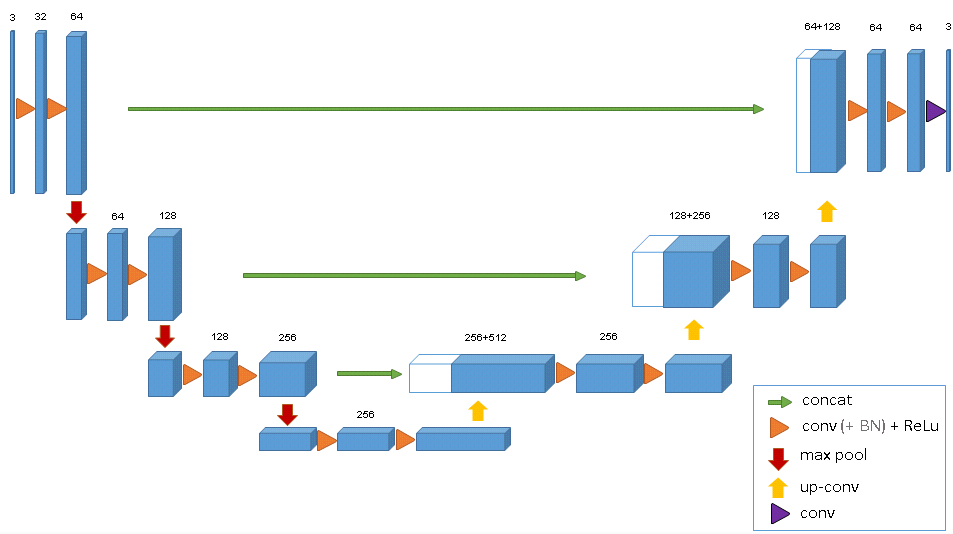
The Convolutional autoencoder U-Net in Keras

Figure 1‑0‑1: The architecture of the image segmentation neural network U-net [4]

The U-net architecture consists of a contracting encoder that analyzes the whole image and a successive expanding decoder to produce a full-resolution segmentation [5]. The network relies on the strong use of data augmentation to use the available annotated samples more efficiently. Such a network can be trained end-to-end from very few images. Here, we will use U-net for denoising.

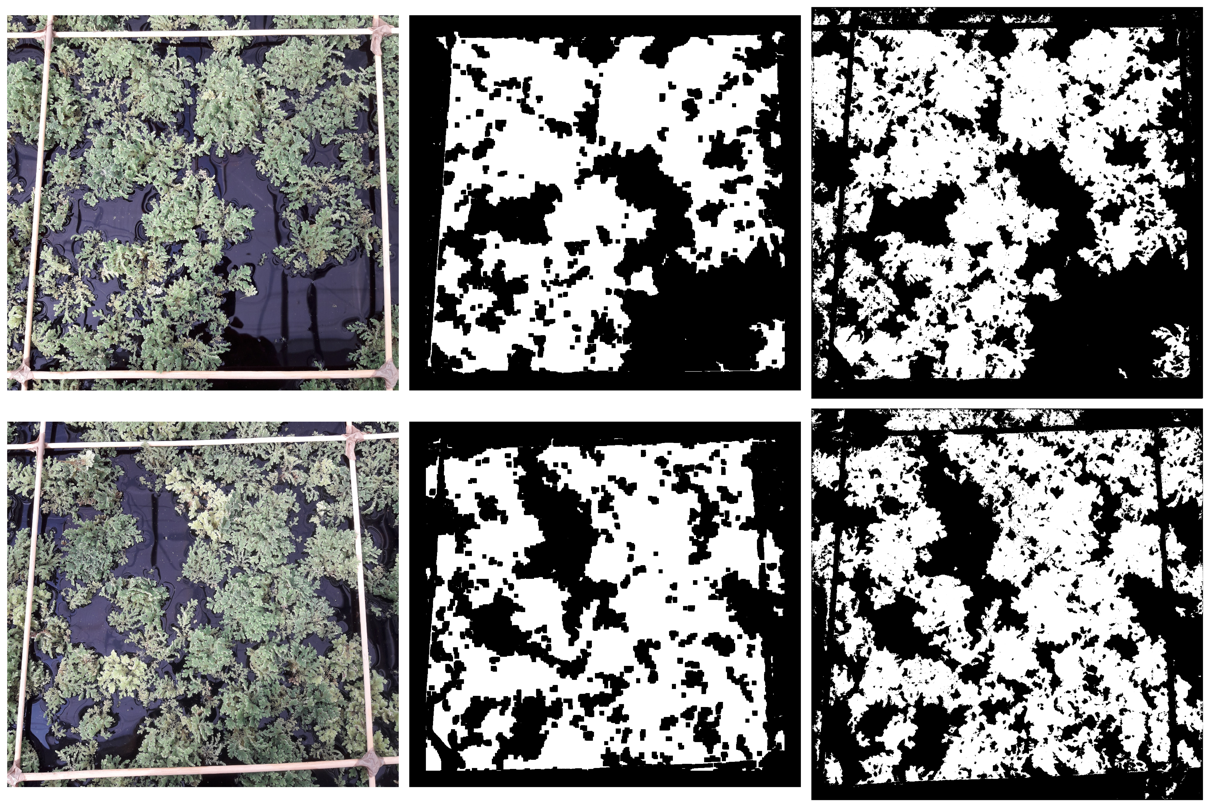


Figure 1‑0‑2: From raw input, to weak labelling, to strong labelling.

Because of the information propagation through the bottleneck of the network, the learnable information is limited. This propagation is optimized with regards to a loss function. By selecting an appropriate loss function that will 1) not punish the network for forgetting information that can’t be extracted from the original image, and 2) reward the network for sharp edges and fully connected areas.

An advantage of the Keras implementation of U-Net is that we can leave the input size undefined: this allows us to train on small images and generalize the model to larger images.

# Incremental Improvement

Now that we have stronger labels for the original dataset, can we update our model with these new labels? Does that actually lead to a better segmentation?

Answer: yes.