User guide for DeepFLaSH:

Our cloud-based deep learning pipeline “DeepFLaSH” is dedicated especially to researchers from the life science community who might have never used machine learning before. We are therefore happy for any kind of feedback from you on how we can further improve DeepFLaSH, for example if you are missing certain functions or think that some sections are not intuitive to use. In general, to get an idea of how DeepFLaSH works and how we recommend its use (e.g. training the CNN-model on segmentation maps created by multiple human experts, evaluation with respect to inter-coding reliability) please find our preprint in which we present DeepFLaSH on bioRvix:

https://www.biorxiv.org/content/early/2018/11/19/473199

This step-by-step user guide was created for users that may have never worked with a Jupyter notebook or Google Colab before. Furthermore, to give everybody the possibility to test our pipeline and to become comfortable with DeepFLaSH, even without having an image dataset yourself, we embedded a set of sample images for demonstration purposes directly into our pipeline. This guide describes the use of DeepFLaSH with the aid of this examplatory dataset but also explains how you can use your own image material. Therefore, let´s get started and enjoy creating you own deep learning models! ☺

Link to our Github repository:

https://github.com/matjesg/DeepFlaSH/

Link to DeepFLaSH on Google Colab:

https://colab.research.google.com/github/matjesg/DeepFLaSH/blob/master/DeepFLaSH.ipynb

Important: You will have to execute several sections of code while you proceed through our pipeline. Since we ensured that you can´t accidently alter the underlying code, the only chance to create an error message is by executing the code sections in the wrong order. For example, you won´t be able to segment any images, if you did not load a CNN-model. Therefore, if you run into an error, we recommend you to refresh the page and to reset the runtime. If the error message persists despite you followed all steps in the correct order, please contact us (deepflash.demo@gmail.com). We are more than happy to help you!

Google Colab provides free of charge access to high performance computing resources like graphics processing units (GPUs) or even tensor processing units (TPUs), which are usually required for machine learning and that are rarely found in life science facilities. All you need to get access to this is to sign-in with a Google-Account (Step 1 inFigure 1). If you don´t have one yet, you can also create one at the sign-in page. Afterwards, move the cursor between the square brackets and a play button will appear in this position (Step 2 in Figure 1). These play buttons execute the respective code sections of the embedded Jupyter notebook.

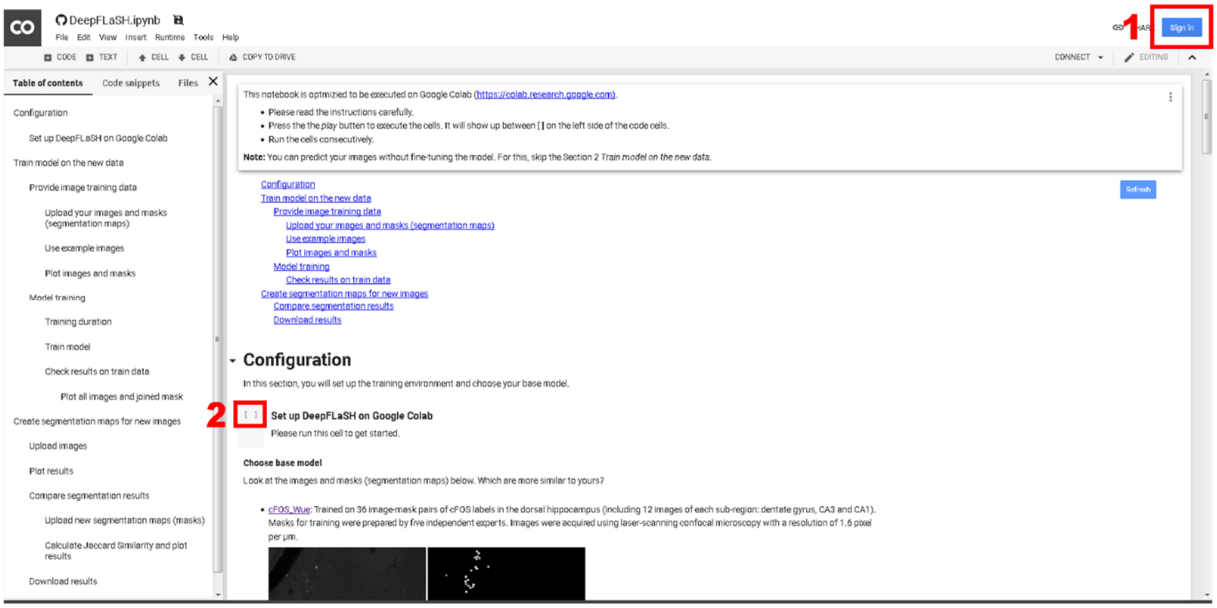

Figure 1: Set up DeepFLaSH on Google Colab.

When you execute this code to set up DeepFLaSH on Google Colab, a “Warning notification” will pop-up (Figure 2). This is simply due to the reason, that the code you are executing was not created by Google but belongs to our source code, which you can find on our Github repository (https://github.com/matjesg/DeepFlaSH/). You can close them and continue to run the code by checking “Reset all runtimes before running”, clicking on “RUN ANYWAY” and “Yes” (Figure 2).

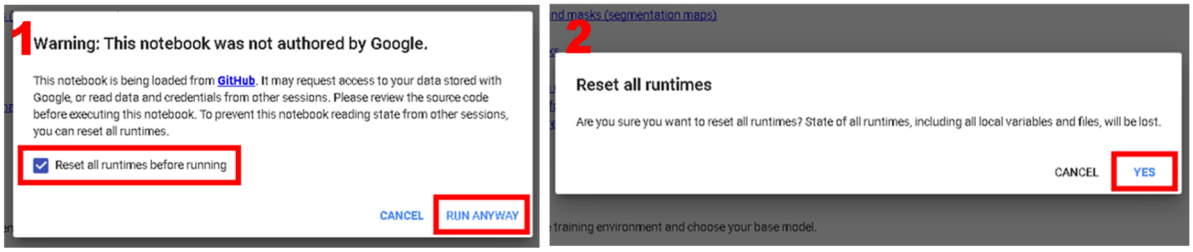

Figure 2: Warning notifications.

Subsequently, Google Colab will reconnect and then execute the code. During execution of a certain code section, the play button will transform in a Stop Button that gives you the possibility to interrupt the execution at any time (usually not needed, see Figure 3). As soon as this is done, you already set up DeepFLaSH on Google Colab.

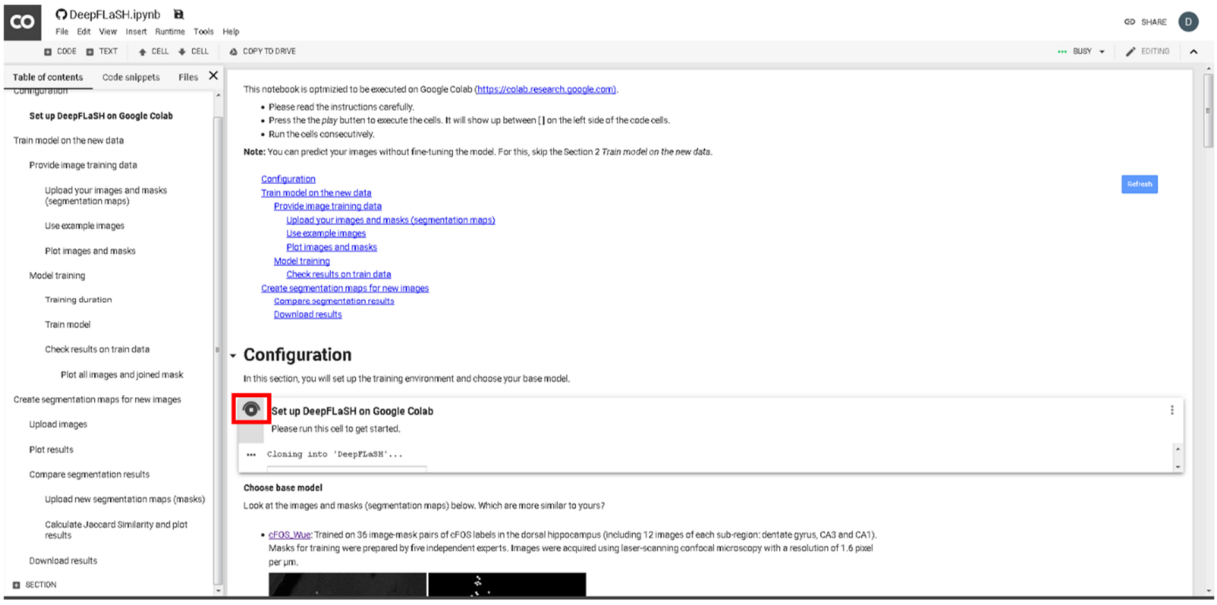

Figure 3: Code being executed.

Now you have the option to train a new model from scratch (no pretrained weights) or to choose a model from our CNN-model library that you would like to adapt to your image data by means of transfer learning. Either way will require pairs of microscopy images and manually created segmentation maps that allow the CNN to learn the correct image feature you would like to segment (Figure 4). The binary segmentation maps (meaning containing only black or white pixels) have to have the same pixel-dimensions as your microscopy image and white pixels correspond to your individual region of interest (ROI) that you want DeepFLaSH to learn and to segment. To make the creation of such segmentation maps as easy as possible for you, we uploaded an ImageJ macro and a short description on how to use it to our GitHub repository.

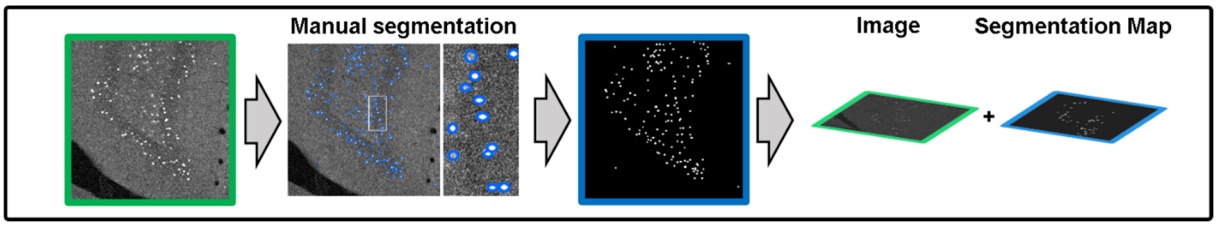

Figure 4: Example of an image-mask-pair.

As there is no ground truth for such an analysis, we recommend you to use segmentation maps created by multiple human experts on base of the same microscopy images. With respect to the idea of inter-coding reliability, this will result in a higher impact of those ROIs that were annotated by multiple coders during training of the neural network and hence ensure its objectivity.

Somewhat intuitive: you will probably require more training data to train a new model from scratch than simply adapting a pretrained model that already learned to segment a certain image feature. You should therefore base this decision on the similarity of your images to those

displayed in our CNN-model library (Figure 5). For example, if you also have an image dataset that shows fluorescent labels of cFOS in brain slices, you should choose the most similar of our pretrained cFOS models, rather than the Parv-model or an untrained model. However, this is not limited to the particular label “cFOS”, but we surmise that this would be the appropriate choice for several nuclear labelings, like for example p-CREB. Therefore we want to encourage you to also try different options to see what works best for your particular image dataset.

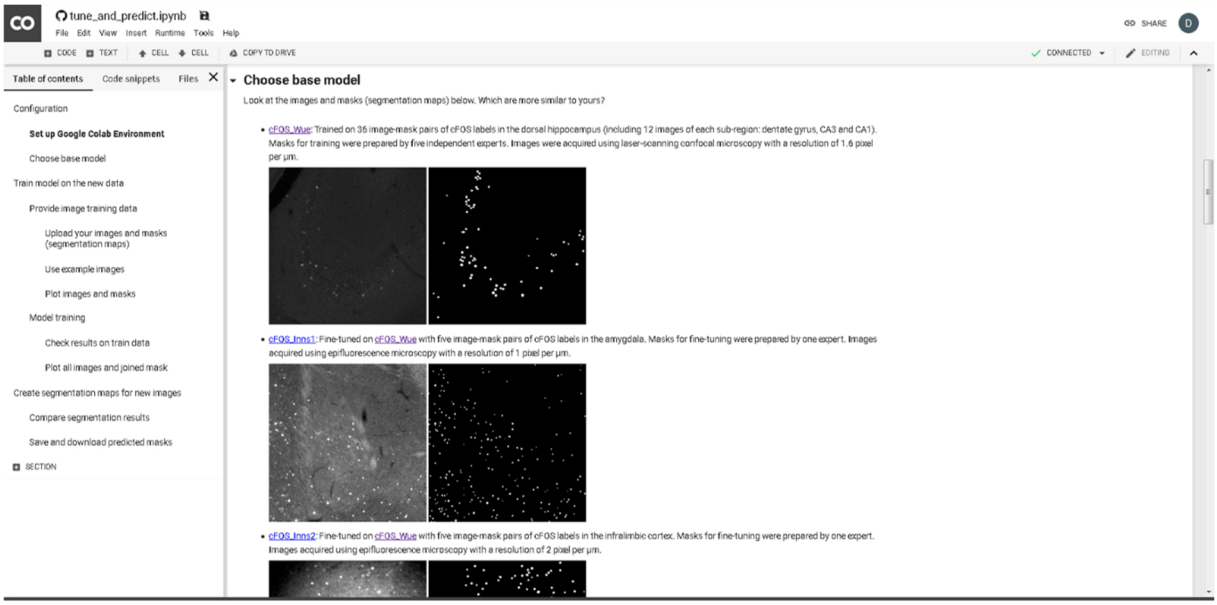

Figure 5: CNN-model library.

To choose the base model (new or pretrained), simply select the respective option from the dropdown menu and download these weights by executing the code

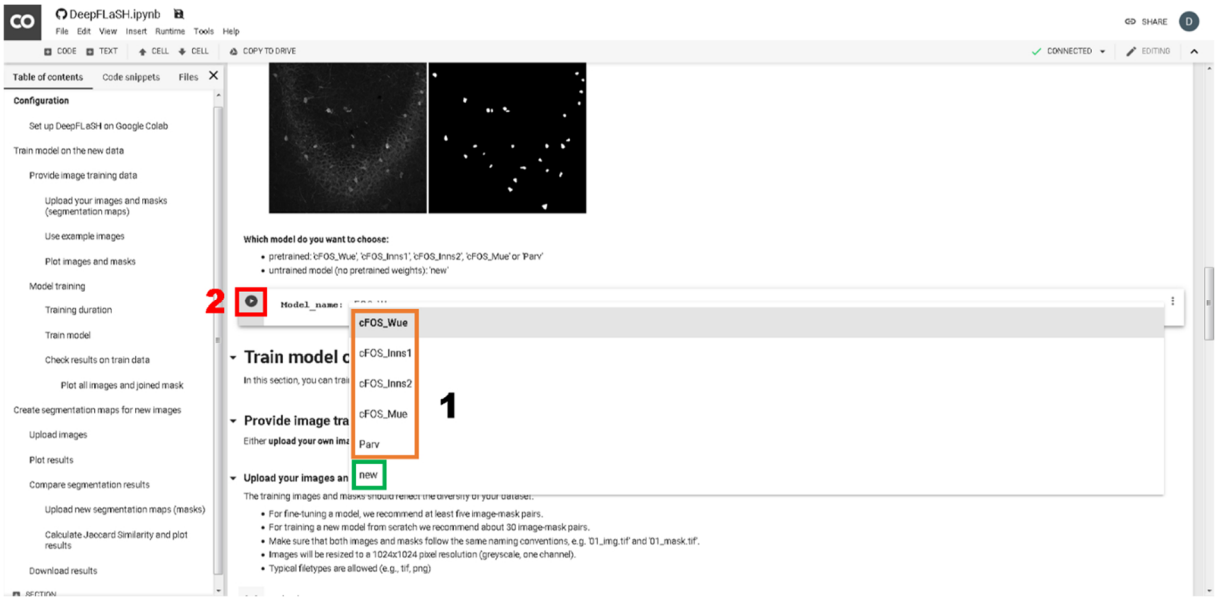

Figure 6: Select a new or a pretrained model as base model.

Subsequently, you can focus on the training of your individual CNN-model. To make your potentially first-ever experience with a deep learning algorithm as easy as possible, we

embedded a set of image-mask-pairs into the pipeline for demonstration purposes. Upon your first usage of our pipeline, we therefore suggest using this dataset to become comfortable with DeepFLaSH (and we will also use this dataset for this user guide, Figure 7).

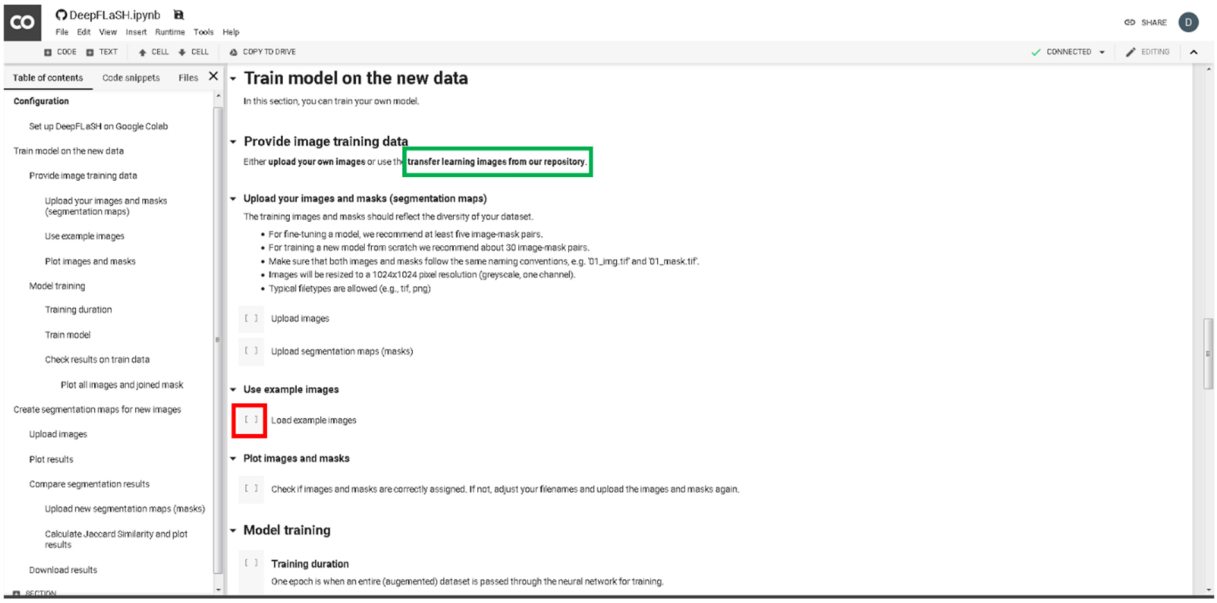

Figure 7: Using our embedded image dataset to become familiar with DeepFLaSH.

To upload your own training dataset, execute the “Upload images” code section (step 1, Figure 8) click on “Durchsuchen..” and upload your image files (step 2, Figure 8). Subsequently, do the same to upload the corresponding segmentation maps (masks). Please follow our guidelines on how to prepare the training dataset (green box,Figure 8). If you followed our recommendation

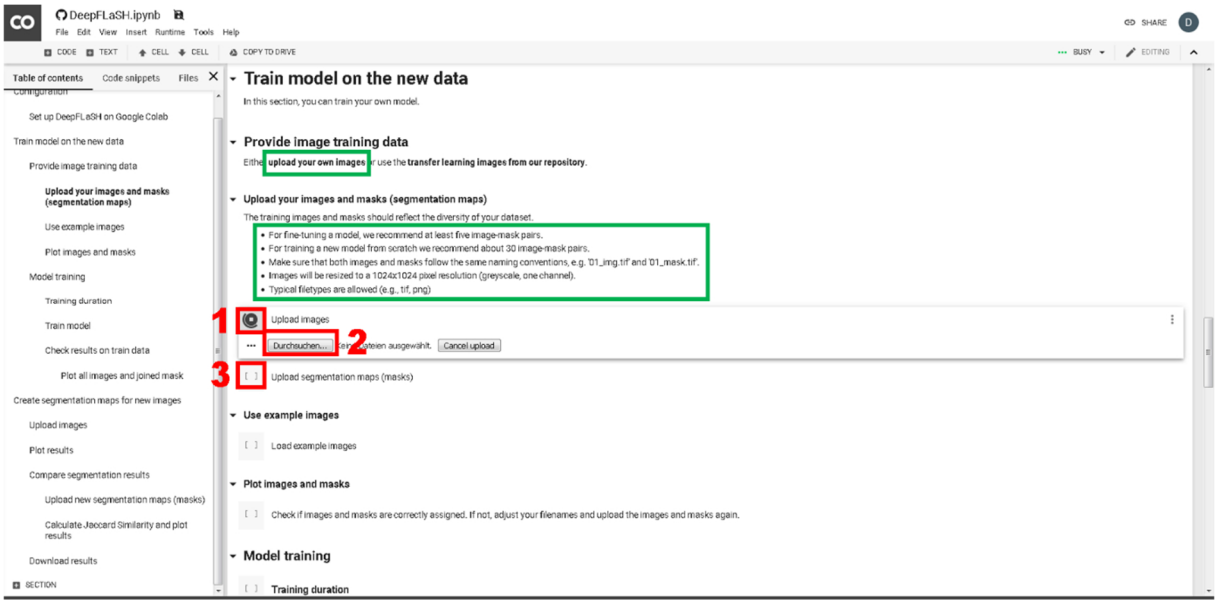

Figure 8: Using your own training dataset to train a CNN-model for your individual segmentation task.

to use masks from multiple human experts for the same image, simply upload the respective microscopy image multiple times. For example, upload the same image as: 0001_img.tif, 0002_img.tif and 0003_img.tif. Then upload the corresponding masks created by three independent human experts as: 0001_msk.tif, 0002_msk.tif and 0003_msk.tif.

After you chose your training dataset (your own or our demonstration dataset), DeepFLaSH gives you the opportunity to check the correct assignment of the image-mask-pairs. After you execute this code section, all images of the training dataset will be plotted with the assigned mask next to it (Figure 9). Please use this step to ensure a correct assignment, since false image pairs will impair the training.

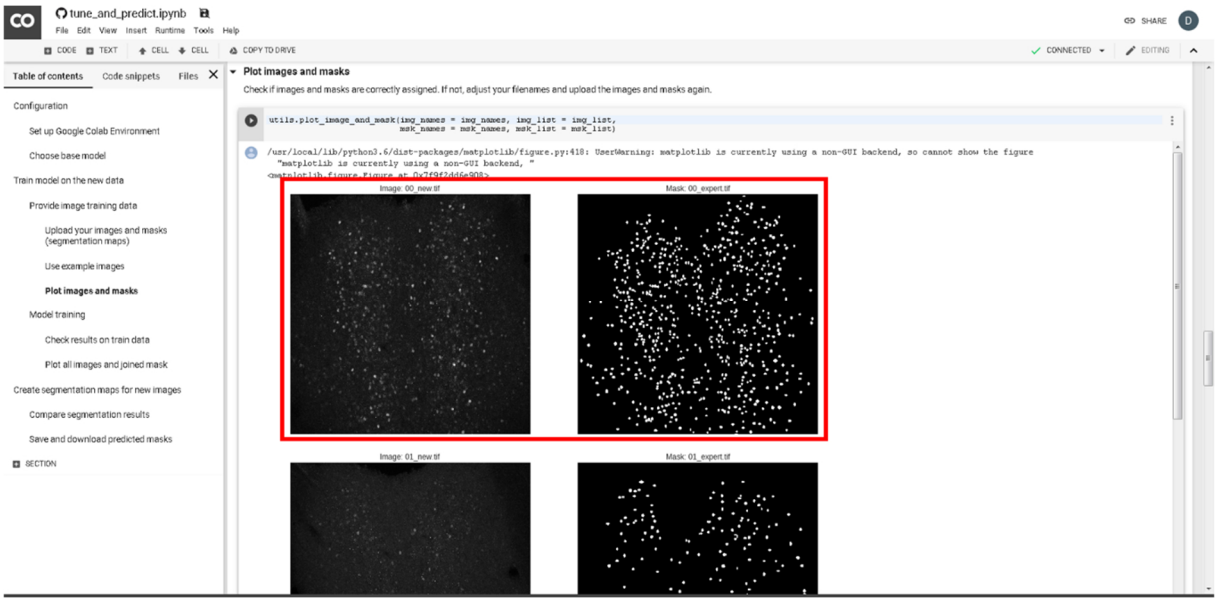

Figure 9: Ensure correct assignment of image-mask pairs

Finally, you are ready to train the convolutional neural network. Select a number of epochs you want the algorithm to train on your training dataset. At this point, it is almost impossible to give you an exact value that will always work, as this highly depends on the quality and the size of the training dataset. Therefore, we can only recommend using at least 50 epochs for fine-tuning a pretrained model to your individual dataset and at least 100 if you want to train a new model from scratch. However, please be aware that training of a deep learning algorithm is a stochastic process in which the algorithm tries to optimize its prediction to match the masks you provided. Therefore, the CNN won´t always become better with each epoch, despite it is more likely to do so. In addition, after too many epochs the algorithm will be overfitted to the training data. In simple words: the algorithm now knows your training data by heart and found a way to match the masks as perfectly as possible, without learning the fluorescent features themselves. Consequently, if you use an overfitted algorithm on new data, it won´t be able to perform the desired segmentation task. This makes it extremely important to validate the model also on images other than those used for training - but don´t worry, we embedded also this feature in DeepFLaSH ;-)

Adjust the number of training epochs by using the sliding tool and confirm this number by executing the code section (steps 1 and 2 in Figure 10). Subsequently, start training of the model by executing the “Train model” section (step 3 in Figure 10).

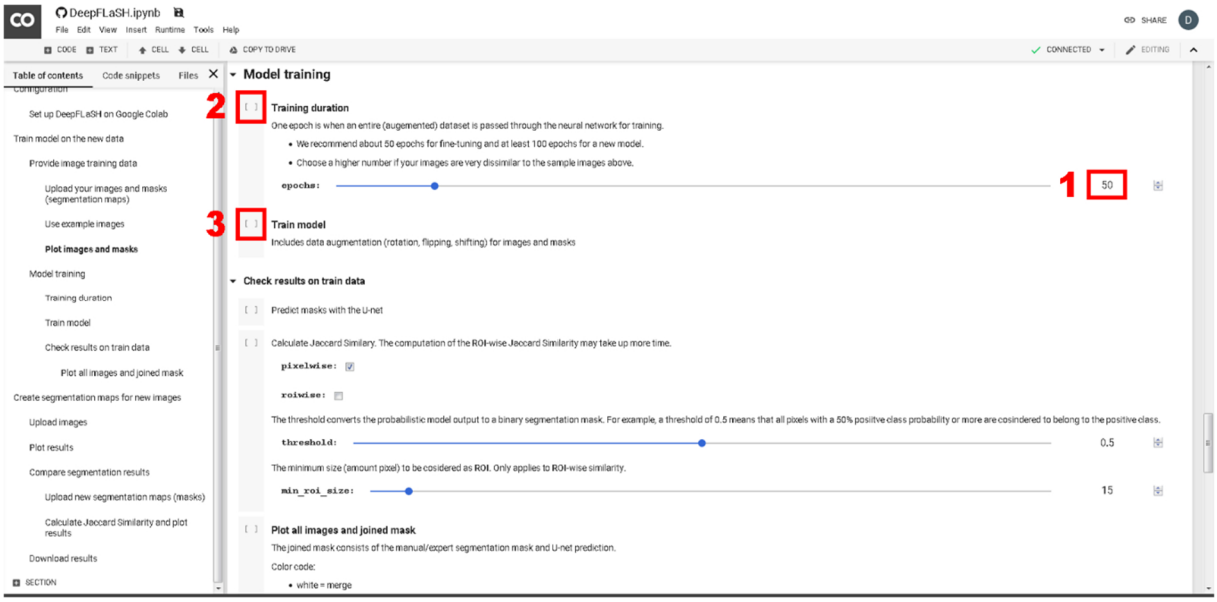

Figure 10: Training a CNN-model.

Despite you have free of charge access to the high performance computing resources of Google Colab, you are still training an enormously complex algorithm. Therefore, please be aware that this process might take some time, depending on the amount of epochs and the size of your training dataset. You will be able to estimate the training duration if you look at the duration it takes for a single epoch (red box, Figure 11). You can also keep track of the current performance of the model after each epoch (green box, Figure 11).

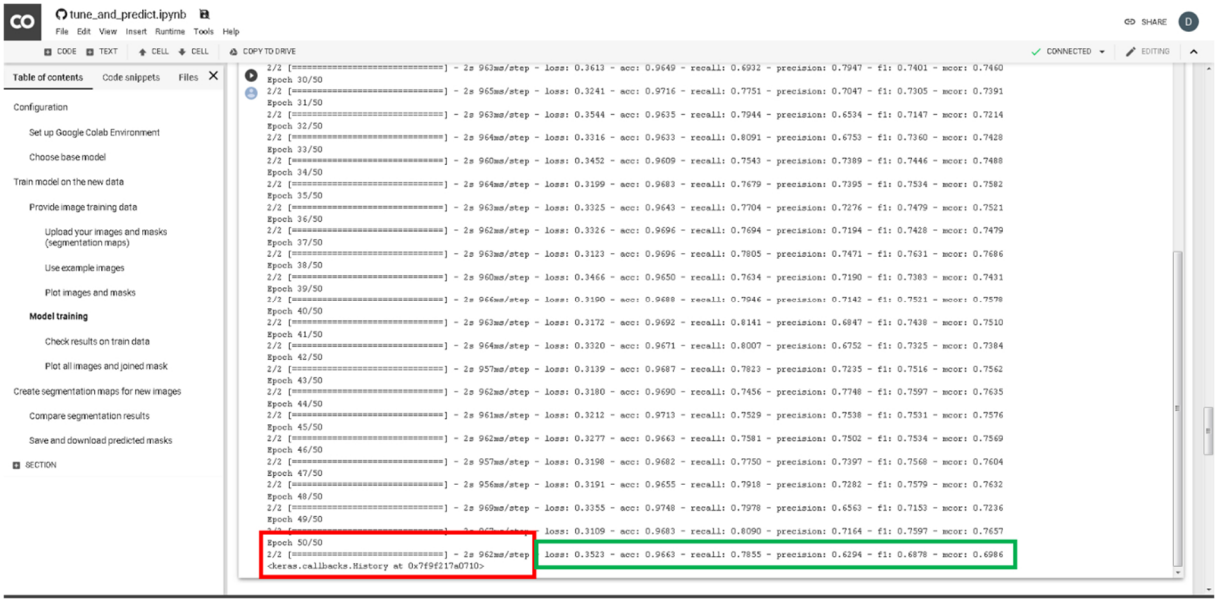

Figure 11: Informations displayed during model training.

Once the training is completed, you can start to evaluate the performance of the model. As an initial validation, we embedded the calculation of the Jaccard similarity (both on ROI and on pixel-level) between the provided segmentation map and the prediction of the trained model. For this, it is important to know that the prediction of the CNN-model is probabilistic. For each pixel, the CNN-model outputs a probability of this pixel belonging to the positive class (= ROI). This probability can range from 0 (absolutely sure this pixel is not part of a ROI) to 1 (fully sure this pixel belongs to a ROI). However, to perform subsequent data analysis, we need a binary

mask and therefore have to convert the probabilistic prediction into a binary segmentation map. To do this, you can use a threshold to take only those pixels into account whose probability of belonging to a ROI exceeds that threshold. In other words, you can decide how stringent you want the algorithm to be. By default, this threshold is set to a value of 0.5. In our study, for example, we used a threshold of 0.95 to decrease the level of uncertainty to a minimum.

For initial evaluation on the training dataset, you have to predict the segmentation masks of the training images with the model you just trained (step 1, Figure 12). Subsequently, you can calculate the Jaccard similarities between the segmentations of the CNN and those you provided. This can be done either on pixel-level or on ROI-level by checking the respective option (step 2, Figure 12). In both cases, you have to specify the threshold to convert the probabilistic CNN output into a binary segmentation map. Only for Jaccard-similarity on ROI-level you can select a minimum size (in pixel) you would expect these ROIs to have (step 3, Figure 12). To estimate this value, we recommend you to use the human expert segmentation maps you created for the training dataset and measure the area of these ROIs. While the calculation of the Jaccard similarity on pixel-level will be quite fast, be aware that the computation of the Jaccard-similarity on ROI-level might take a while.

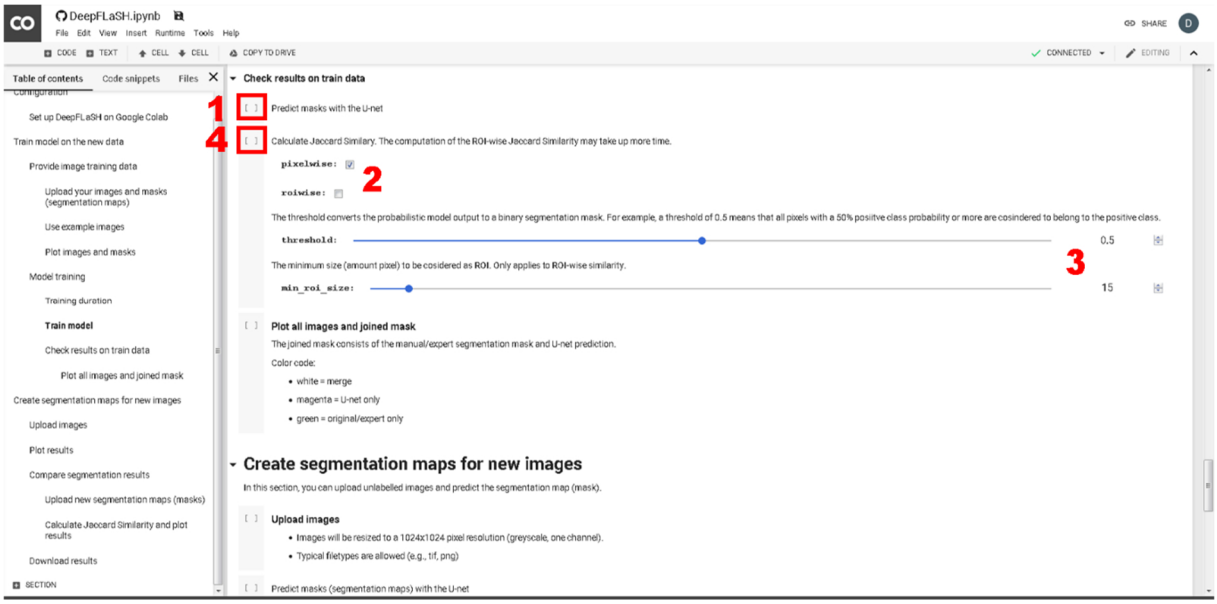

Figure 12: Initial evaluation on base of the training dataset – calculate Jaccard similarity.

When these calculations are done, you can also visually inspect the segmentation performance of the CNN-model. To do so, please execute the code section “Plot all images and joined mask” (Figure 13). DeepFLaSH will now plot the microscopy image and the overlay of the two segmentation maps (expert and CNN) next to each other (Figure 13). The color code of these intersection maps is the following: magenta: pixels solely annotated by the CNN-model; green: pixels exclusively annotated by the human expert; white: pixels annotated by both.

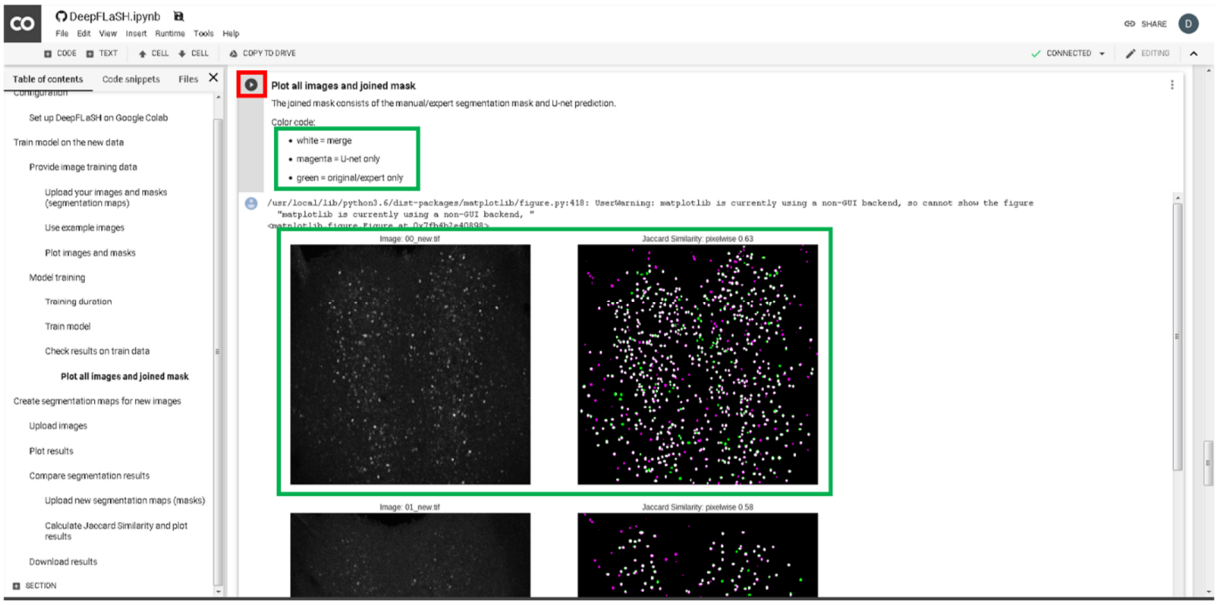

Figure 13: Initial evaluation on base of the training dataset – visual assessment.

When you are happy with the performance of the trained CNN-model, you have to test it on images, which it never saw before, to ensure that the model is not overfitted to your training dataset but really learned the desired fluorescent feature instead. Therefore, please upload the new images (that show of course the same fluorescent labeling you just trained the CNN to segment; step 1 in Figure 14). Then use the trained CNN-model to predict the segmentation maps and plot the results for visual assessment (steps 2 and 3 in Figure 14).

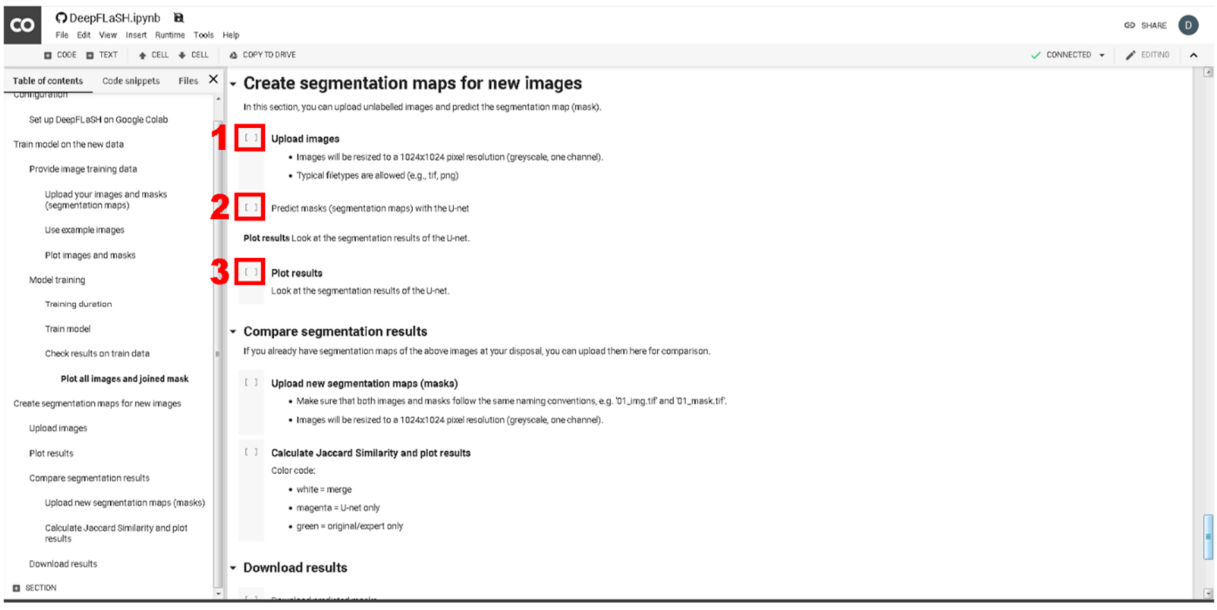

Figure 14: Segment new images with your trained CNN-model.

For a proper validation, however, we highly recommend you to upload another set of human expert-based segmentation maps for this new set of images (step 1 in Figure 15). You can then again compute the Jaccard-similarity for these new images and plot the results as in Figure 13 (step 2 in Figure 15).

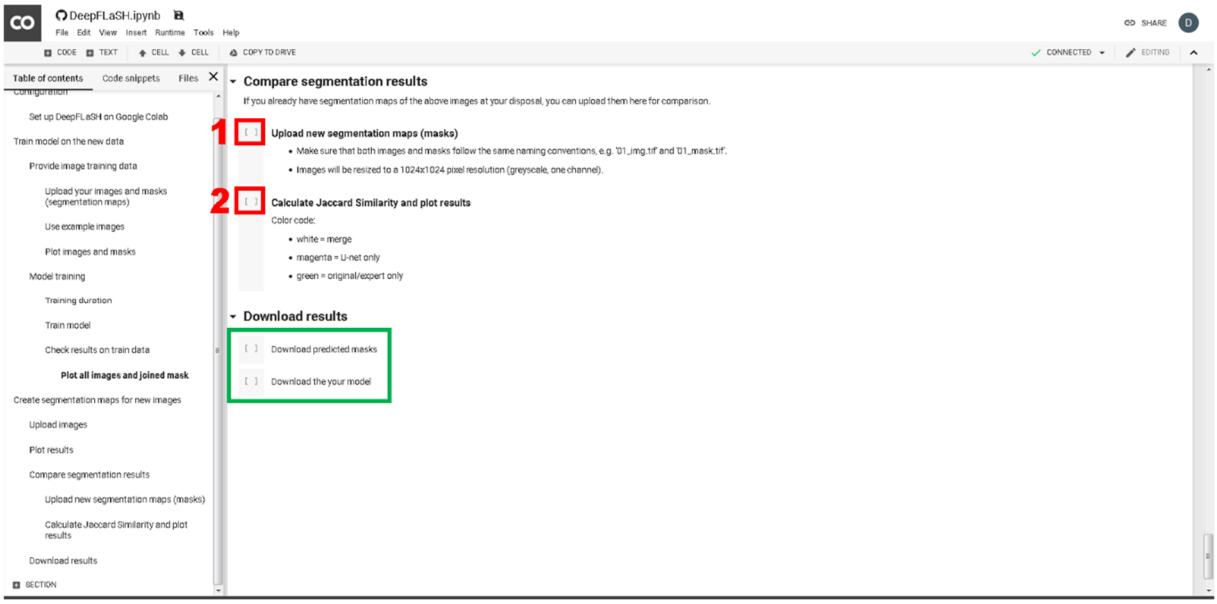

Figure 15: Evaluation of the trained CNN-model on new images and download of the trained CNN-model and the CNN-based segmentation maps.

Ultimately, you can download the CNN-model you trained so that your evaluated model is not lost. Please bare with us, that the function to upload your trained CNN-model into our CNN-model library is not yet available. We are not entirely sure which would be the best way to go here and first want to collect some feedback from you as community and also from the reviewers of our paper. But eventually, this function will become available! ☺

You also have the option to download the segmentation maps to use them for subsequent image data analysis. We are currently also working on a solution to embed this data analysis feature directly into our pipeline, so stay tuned for further updates.

We really hope that DeepFLaSH will make your life easier by bringing the power of deep learning algorithms for image feature analysis as a free of charge, open-source web tool basically into every lab of the life science community. We are happy for every feedback we get to further improve our pipeline or simply to hear that you enjoy using it! ☺