

# Fluffy-guide Guide

Guide to this [Github repository](#).

<b>0. Introduction</b>	<b>2</b>
What is deep learning?	2
What is my workflow doing?	2
The command line isn't a pick-up line for computers...	2
What should I do if things fail?	3
<b>1. Setting up the environment</b>	<b>4</b>
Code	4
Fiji	4
Conda	4
Google Colab	5
<b>2. Generating labeled data</b>	<b>6</b>
File setup	6
File annotation	7
File post-processing	8
<b>3. Training and testing a model</b>	<b>10</b>
<b>4. Using a pre-trained model</b>	<b>11</b>

# 0. Introduction

This guide will provide you with enough information to use and train a fluffily certified model for biomedical image segmentation. Read everything carefully and if you find things to be unclear or outdated let me know by mail – [bastian.eichenberger\(at\)fmi.ch](mailto:bastian.eichenberger(at)fmi.ch).

## What is deep learning?

I might add more detail here. For now, if you're interested you can read this [introduction](#), this [paper](#) or this [video](#).

## What is my workflow doing?

The entirety of my workflow is divided into four main steps:

### 1. Setting up the environment

Initially, the programming and application environment must be set up. This is important as programming applications constantly change. A piece of code is only guaranteed to run if the same dependency versions (code written and published by other people) are used.

### 2. Generating labeled data

For a deep learning model to train it needs data. Ideally the more the better. As some wise old man once said – “garbage in, garbage out” – the higher quality your input is, the better your output will be. So please don't rush this step and take your time. The number of images depends on how variable the images to predict will be. For example, if you try to segment a pattern in a high throughput assay which always looks the same you will need significantly less images than if you try to predict DAPI stained nuclei across multiple microscopes, magnifications, and samples. A good starting point for simpler models is 20-30 images.

### 3. Training and testing a model

This step automatically trains and tests a model. There is a lot to be said here and I will add more information soon.

### 4. Using a pre-trained model

Quite often there might already be a model available to solve your problem. In that case one simply has to use the model on the data to be analyzed.

## The command line isn't a pick-up line for computers...

All subsequent code has to be executed in the command line. Below is a short tutorial to get you going. Note the commands are only for Unix systems (Linux / Mac). All windows users... Please consider your choices and use a real operating system.

To access the command line open the application “Terminal”. Type the commands below in the text field and execute them by simply pressing [enter].

A directory is nothing but a folder. All you need to know is how to find which directory you are in and how to change the current directory. There are three commands you will use:

```
pwd  
ls  
cd
```

- “pwd” – small P, small W, and small D. This stands for “present working directory” and tells you the complete path of your directory.
- “ls” – small L and small S. This stands for “list” and lists all files in the current directory. Use this to find out what directories are available for you to go to
- “cd” – small “C and small D. This stands for “change directory” and is used to navigate to other directories. There are two directions of travel:
  - “cd ..” – Upwards / out of the current directory
  - “cd folder\_xy” – Into the specified folder, here “folder\_xy”

Always use the complete path given by “pwd” in the commands below. This ensures that no mixups between two equally named directories can happen. Watch this [video](#) for a more visual experience.

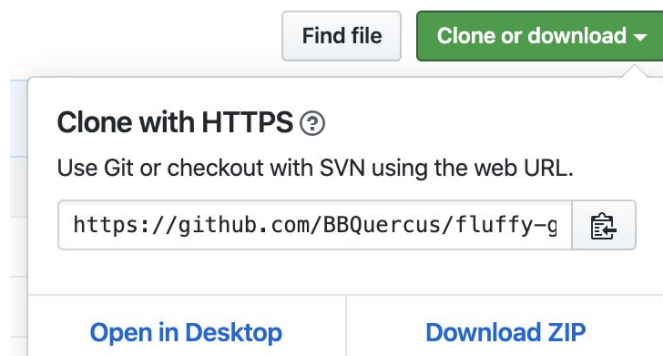
## What should I do if things fail?

Don’t panic. There might still be some undetected bugs. Try to solve the problem **on your own** for at least 15 minutes by reading and understanding what went wrong. Also don’t forget to try turning it on and off. If things haven’t solved themselves feel free to contact me at the mail address above.

# 1. Setting up the environment

## Code

All code that will be used is found in the Github repository @BBQuercus/fluffy-guide. Download its contents and unzip.



Practice your command line skills by navigating to the just downloaded repository.

## Fiji

To annotate data we will need the free image processing program Fiji. Please download and install the latest version for your operating system [here](#).

## Conda

Conda is a package manager for python. We will use it to install all packages needed to run the workflow. Please follow the graphical installation manual for your operating system [here](#). To test the installation type "conda" in the command line. If you don't get an error everything worked as intended. To install the exact packages that were used during development please run:

```
# From within the fluffy-guide directory
conda env create -f environment.yml
```

Follow the installation procedure and you should have a new "environment" (set of python packages) called "fluffy-guide". From now on, every time you want to run something of this workflow run:

```
# From anywhere  
conda activate fluffy-guide
```

This activates the environment meaning you have all tools required.

## Google Colab

This only applies if you have no other access to GPUs and only want to train the simplest of models on your own. If you just want to use a pretrained model, your local computer should be sufficient.

## 2. Generating labeled data

Note – the ALL CAPS words below cannot be used as are but must be changed to suit your specific situation. The names are selected to make sense if read in context.

### File setup

Before we get to labeling images, we need to set up a proper file structure. Please run the following script in the command line after selecting a base directory (e.g. the Desktop) and the folder name (e.g. nucleus\_labeling).

```
# From within the fluffy-guide directory
cd src/data/
python make_directories.py

# Path to the base directory: PATH/TO/BASE/DIRECTORY
# Folder name: NAME_OF_FOLDER
```

Once run, you should have the basic file structure at your specified location. Add all images that you want to label to the “Raw” directory. Only use one of the supported file types: ‘.stk’, ‘.czi’, ‘.tiff’, ‘.tif’, ‘.jpeg’, ‘.jpg’ or ‘.png’. We will now convert all files into a uniform format which will provide the basis to start labeling. As the model which we will create only takes 2D input, images are converted to 2D after running the following command. Specify whether your images in the “Raw” directory are Z-stacks (automatic maximum projection) or time-lapses (only one time frame selected). Alternatively, if your images are already 2D you can simply leave out the specification. Please run:

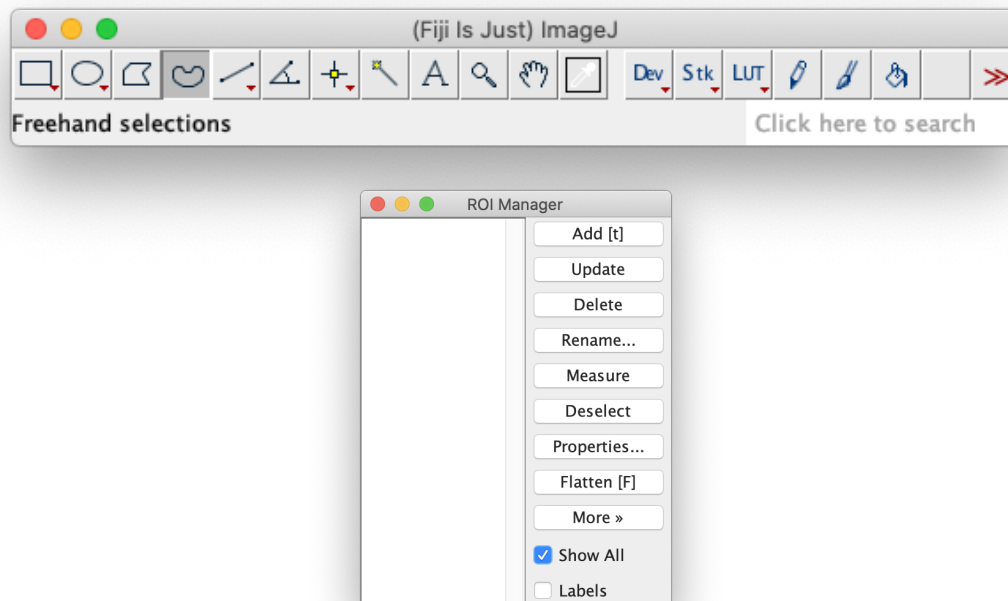
```
# From within the fluffy-guide directory
cd src/data/
python make_labelling.py

# Path to the base directory: PATH/TO/BASE/DIRECTORY
# Timelapse [y/N]: CHOOSE
# ZStack [y/N]: CHOOSE
```

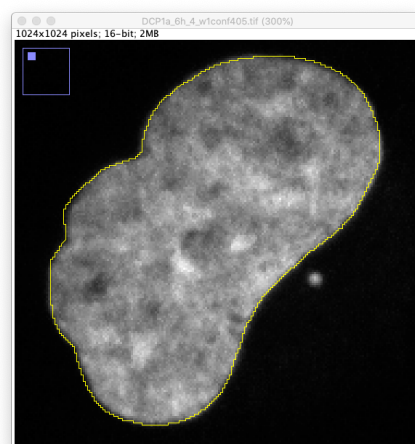
This just converted all files in the “Raw” directory to the 2D images of the ‘.tif’ file format inside the “Labelling” directory.

## File annotation

To annotate images, we will use Fiji. Select the “Freehand selections tool”. Furthermore, open up the “ROI Manager” which can be found under “Analyze > Tools > ROI Manager”. Select the “Show All” option in the ROI Manager. Your setup should now look like this.



Open the first image from within the “Labeling/IMAGE\_NAME/images/” directory and start drawing around the border of one object in your image. Here is an example of a nucleus.



After circelling one object click “Add” or [t] to add the annotated object to the ROI manager. An item with a numeric name should appear in the ROI manager. Continue circelling and adding the objects until all annotations have been added to the manager. Save all annotations by selecting “More > Save...” in the ROI manager. Navigate to the directory

"Labeling/IMAGE\_NAME/" in the just opened popup window. Use the default name "RoiSet.zip" select "Save". To clear the cache of the ROI Manager select all annotations and press "Delete". Repeat the same procedure until all your images have been annotated.

## File post-processing

If you followed the steps above you should now have "RoiSet.zip" files in all "Labeling/IMAGE\_NAMES/" directories corresponding to that image. We will now use a Fiji macro to convert every set of annotations into label masks. Open up the "make\_masks.ijm" file found in the "src/data/" directory inside fluffy-guide by dragging it to the Fiji menu bar. At the very top of the document change "root" to your "Labeling" directory. Execute the macro by selecting "Run" right below the document.

All "Labeling/IMAGE\_NAME/masks/" directories should now contain a bunch of files corresponding to each annotation labeled "mask\_NUMBER.png". Check if your file structure looks the same:

```
# Inside the Processed directory
├── File_Name_1
│   ├── RoiSet.zip
│   ├── images
│   │   └── File_Name_1.tif
│   └── masks
│       ├── mask_0.png
│       ├── mask_...png
│       └── mask_n.png
├── File_Name_2
│   ├── RoiSet.zip
│   ├── images
│   │   └── File_Name_2.tif
│   └── masks
│       ├── mask_0.png
│       ├── mask_...png
│       └── mask_n.png
└── etc.
```

If this is the case, the last step is to merge all of these annotations into a single label map. You can do this by running:

```
# From within the fluffy-guide directory
cd src/data/
python make_label_maps.py

# Path to base directory: PATH/TO/BASE/DIRECTORY
```



You should be greeted by a populated “Processed” directory. Inside, the image files and merged label maps are within the corresponding folders.

### 3. Training and testing a model

“Huston we have a problem” – Training a model, especially if the process is fully automated, takes a lot of computational power. To avoid waiting forever for training to finish one has to use GPUs. Unfortunately setting up GPUs isn’t as straightforward as one could hope for. Currently the easiest option is to speak to contact me once you have prepared enough labeled training data.

The automated training and testing process consists of the following steps:

1. Bayesian probability based hyperparameter optimization

No set of training data is equal. Similarly, there is no one-size-fits-all neural network. Instead, a variety of parameters (hyperparameters) have to be tweaked and tuned to find suitable ones for the problem at hand. Bayesian probability improves this process by iteratively searching across a hyperparameter search space. At the end of this process one should have somewhat suitable hyperparameters for the following steps.

2. Cross validation

This step is not as important as the others but gives us a good feeling on how well the model generalizes (performs on other data). We try to validate the model by selecting different sets of training and validation data to train new models on. To learn more about this procedure you can read up [here](#).

3. Training one final model

Using the found hyperparameters and enough confidence provided by cross validation, we train a final model. This model is trained for longer to maximize the capability of the model to learn important features. This training step is performed on the entirety of training and validation data. The testing data is now used as validation to minimize overfitting.

4. Training multiple final models

Lastly, we have the option to train multiple models on the same dataset. Due to random weight initialization and floating point inaccuracies we will never get the exact same model. Therefore, we can pool together predictions from multiple models to increase performance.

Two main types are available – “binary” and “categorical”. The binary model outputs a True/False mask telling us if an object is there or not (e.g. for granules). The categorical model can be used to predict instances (individual objects) by also predicting object borders which can be used to separate nearby masks (e.g. for nuclei). Start the training regiment by running:

Note – this will change once the pipeline is complete.

```
# From within the fluffy-guide directory
cd src/models/
python train_model.py \
    --data_dir='PATH/TO/DATA/DIRECTORY' \
    # Select if binary or categorical
    --model_type='TYPE' \
```

```
--name='NAME_OF_MODEL'
```

## 4. Using a pre-trained model

A bunch of pretrained models can be found [here](#).

```
# From within the fluffy-guide directory
cd src/models/
python predict_model.py

# Model file: PATH/TO/MODEL_FILE.h5
# Image file/folder: PATH/TO/FOLDER/OR/IMAGE.png
```