Table of content

Reference to the code

If the code provided here or modifications of it are used for scientific publications please refer to it as published in: U. Hohmann et al,* “Assessment of Neuronal Damage in Brain Slice Cultures Using Machine Learning Based on Spatial Features”*, Frontiers in Neuroscience, 2021

0. General remarks

The code provided here contains the final model described in U. Hohmann et al, “Assessment of Neuronal Damage in Brain Slice Cultures Using Machine Learning Based on Spatial Features”, Frontiers in Neuroscience, 2021.

It assumes that the folder and subfolder structure of the provided files is preserved, as some references between modules are present.

Exact dependencies and short explanations of the provided code and its output for all necessary .m files are provided in the first lines of the code as comments. Below a section containing variable parameters of the respective code is given, if no such part exists no parameters can be set.

1. Contained files and folders

The following folders are provided. Please denote that the contained folders and subfolders and files within need to be stored as provided because scripts refer to each other.

1) Apply Models

2) Classify Objects as TP or FP

3) Feature Generation and Manual Labeling

4) Training

5) Sample OHSC

The folder "Sample OHSC" contains two example OHSC together with manually labeled data and all outputs generated during any here provided step of the development of a machine learning model. This folder also serves as an example for the data structuring when applying the provided or new models.

2. Prerequisites for using the given code

2.1 Needed packages

The code provided was written in MatLab 2021a and relies on the following MatLab and Python packages:

MatLab:

- System Identification Toolbox
- Image Processing Toolbox
- Statistics and Machine Learning Toolbox
- Symbolic Math Toolbox

Python:

- sklearn

2.2 Prerequisites to images

To use the given code several pre-requisites for the images have to be fulfilled. The code in its current form is accepting ".tif", ".png", ".jpg", ".pbm", ".pgm",".ppm" or ".bmp" files that are grayscale. As the code reads any image of the given format in the given folder it should not contain additional unwanted images. Additional sub-folders do not pose obstacles. Image stacks are not supported. For full analysis of images data is expected to be ordered as follows: *One main folder, containing sub-folders with the single images of brain slices, from different depth.* See the "Sample OHSC" folder for illustration of ordering and image composition.

3. Analyze images with a pre-trained model

The provided code allows users to apply a pre-trained model assess neuronal damage. For starting the procedure open the ApplyModel.m file in the "ApplyModels" folder with MatLab. The "ApplyModels" folder contains two more subfolders with variable content: "Classify TP and FP objects" and "Trained Models". "Trained Models" contains the number of used principle components and the coefficients of the principal component analysis (PCA) (e.g: "Scalings FinalModel MLP All pca99.9%.m"), as well as the trained neuronal network (e.g.: "FinalModel MLP All pca99.9%.p"). If new, custom models are to be used store the number of used features and PCA coefficient as .mat file and the trained model as .p file in this folder (see chapter 6 for instructions). The folder "Classify TP and FP objects" contains the trained support vector machine (SVM) used for differentiating between true positive and false positive objects, as described by U. Hohmann et al, "Assessment of Neuronal Damage in Brain Slice Cultures Using Machine Learning Based on Spatial Features", Frontiers in Neuroscience, 2021. If new, custom models are to be used store them as .mat file in this folder (see chapter 7 for instructions).

Together with the source code two example OHSC (folder: "Sample OHSC") and all results created by training a model and applying it to those images are included.

3.1 Setting parameters

In lines 6-40 various parameters can be declared, including the location of the conda environment for usage of python inside of MatLab as well as what model etc. to be used. As provided parameters are set to the values described in Hohmann et al. *Please denote that the installation folder and environment name for the conda installation likely has to be changed.*

3.2 Parameters used for the provided pre-trained model

For the pre-trained model provided together with the source code, the following settings were used for feature generation:

% standard deviation of gaussian filters

sigma = [1,3,6];

% Top hat filter sizes used:

TopHatSizes = [3,4,5,7,9,13,15,18];

% Entropy filter sizes used:

EntropySizes = [3,5,7,9,13,17];

Values used for generating the Gaussian derivative filters and Laguerre Gaussian polynomials (see "GenerateFilterBank.m" in folder "Feature Generation and Manual Labeling"):

% Derivatives and directions used for the gaussian derivative Filter:

FiltDirGD = [[1,0,0]; [2,0,0]; [0,1,0]; [0,2,0]; [0,0,1]; [0,0,2]; [1,1,0]; [2,2,0]; [1,1,1]; [2,2,2

% Parameters for Laguerre Gaussian polynomials:

% meshgrid size and spacing

Sz = 5;

StepSz = 1;

% maximal parameters for Laguerre Gaussian polynomials (need to be integers)

% additional, currently fixed parameters are defined further below.

lmax = 2;

pmax = 2;

Pixel size of the images in µm: 1px ≈ 0.52 µm.

3.3 Analyzing images and output files

After setting all these parameters the program can be run and will analyze all images. Thereby, it will create a sub-folder called “Classification” within the folder containing OHSC images. It will contain a file called “NumberPIPositives.mat”, containing the number of identified true positive signals and a .png and .fig file being images of an overlay of the maximal intensity projection of the OHSC and the resulting semantic segmentation. Furthermore, a file named “StatsData ModelName.mat” containing the binary image of the segmentation and the morphological characteristics of the detected objects will be created.

4. Manually label new training data for image segmentation

To label new training data use the files provided in the folder “Feature Generation and Manual Labeling”. To classify new OHSC manually either use the “GetTrainingDataTN.m” or “GetTrainingDataTP.m” files for generating a true negative (TN) or true positive (TP) ground truth. Running these scripts allows users to either mark dead neuronal cells (TP) using a point and click interface or marking polygonal regions without significant cell death (TN). For improved accuracy, the interface for assigning true positive signals has the following commands added:

left arrow key = zoom out at current mouse position

right arrow key = zoom in at current mouse position

backspace = remove last point

left mouse click = add point

The “GetTrainingDataTP.m” file creates a “Training TP.mat” file containing the points marked by the user, subsequently segmented regions and a pixel list containing the location of true positive signals. The GetTrainingDataTN.m” file creates a “Training TN.mat”, containing the regions marked by the user and a pixel list containing their location. These files will be stored in the folder(s) containing the OHSC images.

Note: *For “GetTrainingDataTP.m” and “GetTrainingDataTN.m” the scripts need to be pointed to the subfolder containing the actual OHSC images, <u>not</u> to folder one level higher*.

5. Generate features for image segmentation

To generate a new feature space based on previously classified images use the files provided in the folder “Feature Generation and Manual Labeling”. For this particulate part the files “GetTrainingFeatures.m” and “GenerateFilterBank.m” are of importance.

Use the “GenerateFilterBank.m” to generate the Gaussian derivative filters and Laguerre Gaussian polynomials. Parameters used to set the Gaussian derivative order and direction can be set in lines 5-6. Parameters for the Laguerre Gaussian polynomials are set in lines 9-14. This script creates an output file called “FilderBank.m” containing the created filters used for feature generation. The provided code already contains such a file created with the default parameters used by Hohmann et al:

```
% Derivatives and directions used for the gaussian derivative Filter:

FiltDirGD = [[1,0,0]; [2,0,0]; [0,1,0]; [0,2,0]; [0,0,1];  [0,0,2]; [1,1,0]; [2,2,0]; [1,1,1]; [2,2,2

% Parameters for Laguerre Gaussian polynomials:

% meshgrid size and spacing

Sz = 5;

StepSz = 1;

% maximal parameters for Laguerre Gaussian polynomials (need to be integers)

% additional, currently fixed parameters are defined further below.

lmax = 2;

pmax = 2;
```

To actually generate a feature space used for training a new model the “GetTrainingFeatures.m” file is used. To run the file successfully images need to be labeled first (see chapter 4). In lines 17-25 standard deviation for the Gaussian derivative filters and sizes for the entropy and top hat filters are set. As default the following values were used:

```
% Parameters for feature generation: Needs to be identical to the values of the training

% sigma = standard deviation of gaussian filters

sigma =  [1,3,6];

% Top hat filter sizes used:

TopHatSizes = [3,4,5,7,9,13,15,18];
```

% Entropy filter sizes used:

EntropySizes = [3,5,7,9,13,17];

Running this file creates a file called "TrainingFeatures.mat" containing the features for the true positive and true negative data. Depending on the amount of training data and features this file can become very large.

Note: *The "TrainingFeatures.mat" file needs to be copied to the "Training" folder for actual model training using these features. See also next chapter.*

6. Train new segmentation model on labeled images

To train a new model based on the previously generated feature space use the files in the "Training" folder.

The "LeaveOneImOut.m" file uses the feature data to train the MLP model on all but on OHSC and tests on the remaining OHSC for all possible combinations. It creates a data file called "LeaveOneOutData + ModelName.mat", containing the number of true and false positives and true and false negatives.

Similarly, the "TrainFinalModel.m" file trains the MLP model on all OHSC and saves the final model in the .p file "FinalModel MLP All pcaXY%.p" and PCA scalings in a .mat file called "Scalings FinalModel MLP All pcaXY%.mat".

Note: *For applying the newly trained models to new images using "ApplyModel.m" the model and scaling files need to be stored in the folder "Apply Models\Trained Models". Furthermore, the respective model and scaling names have to be given in the "ApplyModel.m" file in the parameter section (lines 6-40).*

7. Train new model to differ between true and false positive objects

The codes provided additionally contain a module for differentiating between true and false positive objects identified by the MLP model. By default a model is provided in the folder"\Apply Models\Classify TP and FP objects". For training a new one, use the scripts in the folder "Classify Objects as TP or FP".

First, run the "GetTPandFP.m" file and point it to the folder containing the subfolders with the images. Afterwards, the script will generate an output file named "TP_FP_Data.mat" containing the morphological data of detected true positive and false positive objects. The output file will be saved in the same folder as the GetTPandFP.m" file.

Note: *"GetTPandFP.m" can only be used, if the "ApplyModel.m" file successfully run. Otherwise, the segmentation data needed for this step is not present.*

Afterwards, run the “TrainSVM.m” file. It creates an output file with the name “sprintf('%s.mat',Modelname)”, containing the model and number of (in-)correctly classified objects obtained using a cross validation strategy.

Note: *If the newly trained model is to be used during the classification of new images the output file needs to be moved to the folder “Apply Models\Classify TP and FP objects” and the respective model name have to be given in the “ApplyModel.m” file in the parameter section (lines 6-40).*

8. License