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: T. Hohmann et al, Evaluation of machine learning models for automatic detection of DNA double strand breaks after irradiation using a γH2AX foci assay, PLOS One, 2020

1. Contained files and folders

The following folders are provided:

1) Application

2) Training

3) Sample Images

The folder “Application” contains 3 “.py” files called AnalyseImage.py, FociDetctImgAnalysis.py and GetFociData.py. FociDetctImgAnalysis.py and GetFociData.py contain helper functions for the main file AnalyseImage.py, which is used for analyzing images with a pre-trained model. Furthermore, this folder contains a sub-folder called “Trained Model”, containing all used scalings and the pre-trained model used in T. Hohmann et al, Evaluation of machine learning models for automatic detection of DNA double strand breaks after irradiation using a γH2AX foci assay, PLOS One, 2020. The model file is called “MLP_Reduced_Trained”.

The folder “Training” contains 4 “.py” files called TrainModels.py, FociDetctImgAnalysis.py, GetBestParams.py and GetFociData.py. FociDetctImgAnalysis.py, GetBestParams.py and GetFociData.py contain helper functions for the main file TrainModels.py, which is used for training models on previously labeled foci images.

The folder “Sample Images” contains 2 identical images of nuclei, foci and manually marked foci, as well as all sub-folders and files created during model-training and application on these two files (see next chapters).

2. Prerequisites for using the given code

2.1 Needed packages

The code provided was written in Python 3 and relies on the following Python packages:

- NumPy
- Pandas
- SciPy
- os
- sys
- pickle
- csv
- scikit-learn package v0.20.3
- sci-kit image package v0.15

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" and ".bmp" files that can either be rgb or grayscale images. 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 correct working of the code, images of foci and the nucleus have to be provided.

3. Analyze foci images with a pre-trained model

The provided code allows users to apply a pre-trained model to their foci images. As the given approach reduces analysis to a region of interest defined via the nuclear region, images showing the nucleus need to be provided. For starting the procedure open the AnalyseImages.py file in the "Application" folder with appropriate software (e.g. Spyder).

Together with the source code two identical example images, their manual labeling, as well as all results created by training a model and applying it to those images are included.

3.1 Setting parameters

In lines 16-21 of the source code you can specify the folders containing the images of the foci (line 17), the nuclei (line 19) and the folder specifying the path to the model data (line 21). In line 23 the exact name of the model file needs to be given. If the provided pre-trained model is used, input the name "MLP_Reduced_Trained". This is also illustrated in Figure 1.

In lines 25-35 specific image analysis parameters are set. In line 27 a minimal size in pixel for nuclei is set. Objects in nuclei images smaller than this threshold get removed. In line 29 the same property is set for the foci images. In lines 32 and 35 the color channel for the nuclear or foci images are set, assuming those images are rgb. The value 0 corresponds to the red channel, 1 to the green channel and 2 to the blue channel. For gray scale images these properties are not used and can be chosen arbitrarily. This is also illustrated in Figure 1.

In lines 37-44 properties and sizes of filters are set. Please refer to "T. Hohmann et al, Evaluation of machine learning models for automatic detection of DNA double strand breaks after irradiation using a γH2AX foci assay, PLOS One, 2020" and the scikit-learn documentation for more information. This is also illustrated in Figure 1.

Line 48 introduces a rescale factor, increasing or reducing image size in each dimension by the given factor. This might be useful to adopt a model to images of different resolution to reach a roughly constant foci size between test and training images. Notably, the higher the rescale factor the higher the computation time.

```
# -*- coding: utf-8 -*-
"""
Created on Fri Mar 29 10:39:10 2019

@author: Tim Hohmann et al. - "Evaluation of machine learning models for
automatic detection of DNA double strand breaks after irradiation using a gH2AX
foci assay", PLOS One, 2020
"""


# main file to analyse new images using a pre trained model

################################################################################
# Parameters and file path that have to be set manually:

# directory containing the foci images:
im_path_foci = "D:\\Sample Images\\foci"
# directory containing the dapi images:
im_path_dapi = "D:\\Sample Images\\dapi"
# path with the information for the trained model:
model_path = "D:\\Sample Images\\rained Models"
# file name of the model
model_name = "MLP_Trained"

# Parameters:
# min area of nucleus
min_area = 4000
# min foci area:
min_area_foci = 16
# color channel of nucleus. 0 = red, 1 = grenn, 2 = blue. for grayscale images
# this value is ignored.
nuc_chan = 2
# color channel of foc. 0 = red, 1 = grenn, 2 = blue. for grayscale images
# this value is ignored.
foci_chan = 1

# specify filter sizes. for best results these should be identical to the ones
# used for model training.
# used filter sizes
filt_range = [2,10,30]
# scaling range for frangi filter
sc_range = list(range(2,11))
#frequency range for gabor filter
freq = [0.08,0.16,0.2]

# adjust image size (should be adjusted to match resolution of training images)
#image rescale factor:
rescale_factor =1.8

```

Figure 1: Sample parameter setting for model application.

3.2 Parameters used for the provided pre-trained image

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

filt_range = [2,10,30]

sc_range = list(range(2,5))

freq = [0.08,0.16,0.2]

Pixelsize of the images in µm: 1px ≈ 0.12 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 “Single Nuclei” within the folder containing the nucleus images. This folder contains binary images showing the nucleus segmentation.

Furthermore, a sub-folder named “Model Name”_Trained_Results will be created containing the foci images – with foci now displayed in green – with the respective classification of each pixel, shown in white. Additionally, a .csv file named “Model Name”_Trained_Results.csv is created which contains the number of pixels covered by foci relative to the nucleus area for each nucleus and image.

4. Train model on labeled images

The provided code allows users to train a model using manually labeled foci images. As the given approach reduces analysis to a region of interest defined via the nuclear region, images showing the nucleus need to be provided. For starting the procedure open the AnalyseImages.py file in the “Application” folder with appropriate software (e.g. Spyder).

Together with the source code two identical example images, their manual labeling, as well as all results created by training a model and applying it to those images are included.

4.1 Setting parameters

In lines 45-50 of the source code you can specify the folders containing the images of the foci (line 46), images of the manually labeled foci (line 48) and the nuclei (line 50). In line 49 the name of the model file is specified, which is later used for savings. This is also illustrated in Figure 2.

In lines 12-22 specific image analysis parameters are set. In line 13 a minimal size in pixel for nuclei is set. Objects in nuclei images smaller than this threshold get removed. In lines 16 and 19 the color channel for the nuclear or foci images are set, assuming those images are rgb. The value 0 corresponds to the red channel, 1 to the green channel and 2 to the blue channel. For gray scale images these properties are not used and can be chosen arbitrarily. In line 22 the color used for the manual markings is defined. As this is fixed to be red, green or blue it is highly recommended to use one of these colors for the markings. This is also illustrated in Figure 2.

Line 27 introduces a rescale factor, increasing or reducing image size in each dimension by the given factor. This property is useful for downscaling images of very high resolution to improve calculation time. Line 34 can also be used to speed up the training process, as the sampling property sets what proportion of the data of each image is used for training. This is especially useful when using many training images to speed up the training process. Notably, both rescaling and sampling generally affect the accuracy of model training negatively and thus these parameters have to be chosen with care.

Another strategy employed here for improving training time is that of dimension reduction. Therefore, linear principal component (PCA) analysis is used. After PCA the obtained principal components are linearly de-correlated and higher order components thus do not add significant "new" information (in the linear PCA sense) and thus can likely be omitted. The parameter in line 31 sets the number of kept principal components by defining the cumulative variance ("information") they contain. Higher numbers keep more components, lower less. Consequently, reducing this value speeds up the training process, but may lead to a loss of information.

In lines 36-44 properties and sizes of filters are set. Please refer to "T. Hohmann et al, Evaluation of machine learning models for automatic detection of DNA double strand breaks after irradiation using a γH2AX foci assay, PLOS One, 2020" and the scikit-learn documentation for more information. This is also illustrated in Figure 2.

```
# -*- coding: utf-8 -*-
"""
Created on Sun Feb 24 19:08:39 2019
@author: Tim Hohmann et al. - "Evaluation of machine learning models for
automatic detection of DNA double strand breaks after irradiation using a gH2AX
foci assay", PLOS One, 2020
"""
# main file for training machine learning models using previously labeled data
################################################################################
# Parameters and file path that have to be set manually:
# Parameters:
# min area of nucleus
min_area = 4000
# color channel of nucleus. 0 = red, 1 = grenn, 2 = blue. for grayscale images
# this value is ignored.
nuc_chan = 2
# color channel of foci. 0 = red, 1 = grenn, 2 = blue. for grayscale images
# this value is ignored.
foci_chan = 1
# color channel of marked image. 0 = red, 1 = grenn, 2 = blue. corresponds to
# the color of the markings in the manually labeled foci images
mark_chan = 0

# adjust image size - might be usefull to save calculation time. needs to be
# identical for foci and nucleus images
# image rescale factor:
rescale_factor = 1.0

# take only those PCA components cumulatively explaining var_max of the variance
# 0<var_max<=1.
var_max = 0.95
# randomly sample a proportion of the training data from each image (0<sampling<=1).
# speeds up training process if smaller than 1
sampling = 1

# used filter sizes
filt_range = [2,3,4,5,8,10,15,20,25,30,35]
# scaling range for frangi filter
sc_range = list(range(2,11))
#frequency range for gabor filter
freq = [0.08,0.10,0.13,0.16,0.2]

# Name used for saving the trained model and related images:
model_name = "MLP"
# directory containing the foci images:
im_path_foci = "D:\\Sample Images\\foci"
# directory containing the labeled images:
im_path_foci_marked = "D:\\Sample Images\\foci marked"
# directory containing the nucleus images:
im_path_dapi = "D:\\Sample Images\\dapi"

```

Figure 2: Sample parameter setting for model training.

4.2 Setting the model

In the current version of the code a multi-layer perceptron model is used for training. To change this go to lines 137 and the following (Figure 3) and change it accordingly. The variable “parameters” describes multiple properties of the model that can be set and changed. In the current version it defines multiple “alpha-values” and “batch sizes”. The values set there are tested using a grid search (line 162/163) to find best suited settings.

Additional code for sample models – all part of the scikit-learn package – are also provided and just need to be uncommented (e.g. random forest in lines 141-146, Figure 3). For more information on available models, their parameters etc. the reader is referred to the scikit-learn documentation.

```
129 ##############################################################################
130 # This part is for model training assuming "get_labeled_data" was ran successfully
131
132 # get trainingsdata:
133 x_train_transformed, y_train, idx, s1, s2, p, mnmx = image_pipeline(x_data,y_data,removed_im = [],var_max = var_max, sampling = sampling)
134
135 # Chose the model to train:
136 # neural network:
137 model =  MLPClassifier(alpha=0.1, batch_size = 2000, learning_rate = "adaptive", \
138                         learning_rate_init = 0.1, max_iter = 300, tol = 10**-4, early_stopping = True)
139 parameters = {'batch_size':[100,500,1000,2000,3000,4000,5000],'alpha':[10**-4,10**-3,10**-2,10**-1,1]}
140
141 # random forest:
142 # clf = RandomForestClassifier(criterion = "entropy",min_weight_fraction_leaf = 0.005, \
143 # n_estimators = 15,max_depth = 50, min_samples_leaf = 10,min_samples_split = 100, n_jobs = -1)
144 # model =  AdaBoostClassifier(base_estimator = clf, n_estimators=5)
145 # parameters = {'base_estimator__min_weight_fraction_leaf':[0.0001,0.001,0.005],\
146 #'base_estimator__n_estimators':[5,10,15,20],'base_estimator__min_samples_leaf':[10,20,100]}
147
148 # complement naive bayes:
149 # clf = ComplementNB(alpha = 0.0, norm = True)
150 # model =  AdaBoostClassifier(base_estimator = clf, n_estimators=15)
151 # parameters = {'base_estimator__alpha': [0,0.01,0.02,0.03,0.04,0.05,0.06], 'base_estimator__norm': [True, False]}
152
153 # support vector machine:
154 # linear svm
155 # clf = LinearSVC(penalty = "l2", loss = "hinge", C = 2, class_weight = "balanced", max_iter = 5000)
156 # model =  AdaBoostClassifier(base_estimator = clf, n_estimators=5,algorithm='SAMME')
157 # parameters = {"base_estimator__C": [0.1,0.3,0.6,1,2,3]}
158
159
160
161 # get best model parameters:
162 clf = GridSearchCV(model, parameters, cv = 3)
163 clf.fit(x_train_transformed, y_train)
164
```

Figure 3: Sample model for model training.

4.3 Train model and output files

After setting all these parameters the program can be run and will analyze all images. First the model will be trained on all but one images and tested on the remaining one. Afterwards, it will create a sub-folder called “Results Model Validation” within the folder containing the foci images, containing classified test images, with identified foci marked in white. Furthermore, an image_statistics.txt file is created, containing the image number, the confusion matrix, mean accuracy of classification, precision and recall for each image tested with the leave-one-out strategy.

As a last step the model is trained on the whole dataset and a folder called “Trained Models” containing a folder “Model Name” is created. Inside the folder all necessary model and scaling data is stored. Notably, the saved model is called “Model_name”_Trained.

5. License

Kommentar [0TM1]: @Faramarz: Wen muss man da eigentlich angeben?