NanoNet Guide

Contents

How to download and install NanoNet

Simply click on the latest NanoNet.exe folder and download it.

NanoNet workflows

NanoNet prompts you to choose between two main workflows: segmentation and feature extraction for one condition (**A**) or machine learning aided classification and feature comparison between different conditions (**B**) (**Figure 1**). For this pipeline we used python scikit-learn packages developed by (Pedregosa et al., 2011)

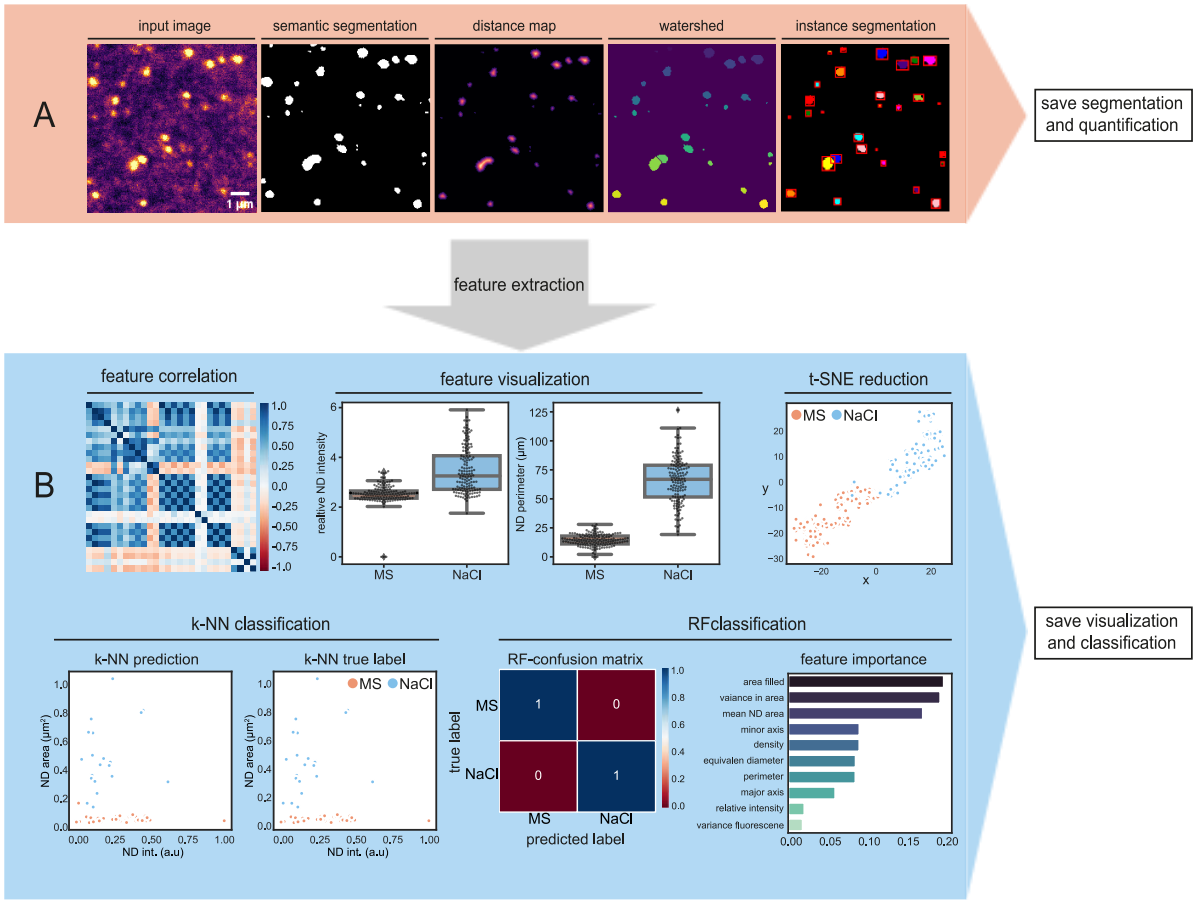

Figure 1 ¦ NanoNet workflows. (**A**) for segmentation and quantification of ND images (from 1 folder, same condition). (**B**) For classification and feature comparison between different images (from ≥2 folders, different conditions).

A. Segmentation and feature extraction:

This part of the NanoNet expects one folder with TIFF images, displaying your regions of interest (ROIs). For certain features (such as Nanodomain density) it is important that your ROIs all have the same size. The app will prompt you to enter the size in pixels. This can be checked in common image analysis tool such as Fiji. As output you will get a results excel sheet with all the features extracted. For feature annotation see (**Table 4**).

1.) Choose your input folder.
2.) Enter your pixel size in microns. (you can check this in any common software for example Image J (**Figure 2**)).
3.) Choose your output folder.

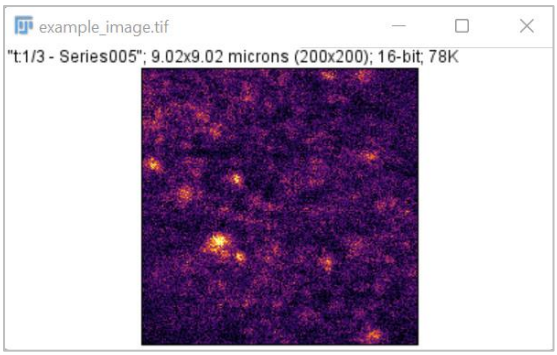

Figure 2 ¦ The pixel size to enter in NanoNet would be 9.02 in this case.

B. Comparison and Classification between conditions:

This part of the NanoNet expects at minimum 2 input folders with TIFF images, displaying your regions of interest (ROIs) you want to compare. For certain features (such as Nanodomain density) it is important that your ROIs all have the same size. The app will prompt you to enter the size in pixels. This can be checked in common image analysis tool such as Fiji. The extracted features of each condition (image folder) will be passed on to train a Random Forest (RF) and a k-NN classifier (k-NN). Additionally you will get Boxplots comparing chosen features between your conditions, a t-SNE map and an inter-feature correlation map. For feature annotation see (**Table 4**).

1.) Choose your folder paths and give each condition a name. You can also choose a colour for each condition which will be used to draw the boxplots and scatterplots.
2.) Order the conditions in the way you want them to appear in the boxplots.
3.) Enter your pixel size in microns. (you can check this in any common software for example Image J (**Figure 2**)).
4.) Set parameters for t-SNE visualization, k-NN - and RF-classification.
5.) Using an 8-fold cross validation the optimal number of k neighbours is search within the range of k you entered before.
6.) Choose features to display as boxplots.
7.) Choose output folder.
8.) Results are displayed and saved. For a detailed list of individual feature importance in the RF-model click on the REPORT button.

Glossary

Train/Test split
Sets how much of your data should be used to train the classifier v.s test the model. The float should be between 0.0 and 1.0 and represents the proportion of the dataset to include in the test split. For example the default 0.25 uses 75% of the data to train the model and 25% to test it. For more information see scikit train_test_split.

Random Forest classifier (RF)

In RF (Breiman, 2001), each tree is built from a sample drawn with replacement from the training set. Simply spoken, the majority vote of all trees decides the prediction of class label based on random subset of input features of size max_features. Due to these two sources of randomness, the RF model has shown to be less prone to overfitting. For more information see scikit RF. As an output, the RF-confusion matrix, was well the impurity based feature importance are given. For selectable parameters see (**Table 1**).

Table 1 ¦ Random Forest parameters (adapted from scikit RF)

N_Estimators	int, default = 1000	The number of trees in the forest.
Maximal tree depth	int, default =110	The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
Minimum samples split	int or float, default = 2	The minimum number of samples required to split an internal node: If int, then consider **min_samples_split** as the minimum number. If float, then **min_samples_split** is a fraction and ceil (min_samples_split * n_samples) are the minimum number of samples for each split.
Minimum leaf samples	int or float, default = 1	The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least **min_samples_leaf** training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

k-nearest neighbors classifier (k-NN)

The k-NN model is a type of instance-based or non-generalized learning model. It computes distances from all features in the training dataset which are used to predict the class of the test datapoints accordingly. For more information see scikit neighbors classification and (Altman, 1991; Fix & Hodges, 1989). A grid search is performed to find the optimal number of k-neighbors and 8-fold cross validation is used. As an output, the balanced accuracy score as well as well as two scatterplots depicting the predicted and the true label classification are given. For selectable parameters see (**Table 2**).

Table 2 ¦ k-NN parameters (adapted from scikit k-NN)

Maximal number of neighbors	int, default = 9	Max. Number of neighbors for gridsearch to find optimal k.

t-distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE is a stochastic non-linear dimensionality reduction algorithm to visualize high-dimensional data in lower dimensional space (here 2D)(van der Maaten & Hinton, 2008). Keep in mind that the choice of parameters greatly influence the visual output. For more information see

scikit t-SNE. A good resource for parameter influence is "How to use t-SNE effectively". The learning rate is set to "auto" according to (Belkina et al., 2019; Kobak & Berens, 2019). As an output, the t-SNE dimensionality reduced data is visualized as scatterplot. For selectable parameters see (**Table 3**).

Table 3 ¦ t-SNE parameters (adapted from scikit t-SNE)

Perplexity	float, default=0	Defines how to balance the focus on local and global features of your data. Higher perplexities = more nearest neighbors, less sensitive to small structure. Lower perplexity = ignores more global information in favour of the local neighborhood. For larger datasets, more points are required to represent the local neighborhood (larger perplexity). Similarly, noisy datasets require larger perplexity values to encompass the local structure. Usually set between 5-50.
N_Iterations	int, default=500	Max. number of iterations is usually high enough and does not need any tuning.

Quantified features

Table 4 ¦ feature annotations

"name"	name of the image.
"area"	area in microns per segmented ND.
"mean_area"	mean area in microns of segmented NDs per image.
"var_area"	variance of ND area within one picture
"density"	density of NDs within one picture as ratio between ND occupied pixels to all pixels.
"density_microns"	density of NDs within one picture in microns.
"intensity"	mean intensity per segmented ND.
"relative_intensity"	ratio of mean intensity per segmented ND to mean intensity of the image.
"mean_intensity"	mean intensity of segmented NDs per image.
"var_intensity"	variance of ND intensity per image.
"max_intensity"	maximum intensity per ND.
"mean_max_intensity"	mean maximum ND intensity per image.
"min_intensity"	minimum intensity per ND.
"mean_min_intensity"	mean minimum intensity of NDs per image.
"area_filled"	area in microns of the ND with all holes filled in.
"mean_area_filled"	mean area in microns of the ND with all the holes filled in per image.
"major_axis_length"	length in microns of the major axis of the ellipse that has the same normalized second central moments as the ND.
"mean_major_axis_length"	mean length in microns of the major axis of the ellipse that has the same normalized second central moments as the NDs per image.
"minor_axis_length"	length in microns of the minor axis of the ellipse that has the same normalized second central moments as the ND.

"mean_minor_axis_length"	mean length in microns of the minor axis of the ellipse that has the same normalized second central moments as the NDs per image.
"eccentricity"	eccentricity of the ellipse that has the same second-moments as the ND. The eccentricity is the ratio of the focal distance (distance between focal points) over the major axis length. The value is in the interval [0, 1). When it is 0, the ellipse becomes a circle.
"mean_eccentricity"	mean of the ND eccentricity per image.
"equivalent_diameter_area"	The diameter in microns of a circle with the same area as the ND.
"mean_equivalent_diameter_area"	mean of the equivalent_diameter_area per image.
"perimeter"	perimeter in microns of ND which approximates the contour as a line through the centers of border pixels using a 4-connectivity.
"mean_perimeter"	mean perimeter per image.
"ND_ID"	label number of segmented ND.
"ND_quantitiy"	sum of all labels (number of segmented NDs) per image.
"var_intensity_image"	intensity variance per image.
"mean_intensity_image"	mean intensity per image.
"mean_intensity_blurred"	mean intensity of gaussian blurred image.
"var_intensity_blurred"	variance intensity of gaussian blurred image.
"SCI"	spatial clustering index per image.

Features which include ROI properties were extracted using scikit regionprops.

References

Altman, N. S. (1991). *An Introduction to Kernel and Nearest Neighbor Nonparametric Regression An Introduction to Kernel and Nearest Neighbor Nonparametric Regression*.

Belkina, A. C., Ciccolella, C. O., Anno, R., Halpert, R., Spidlen, J., & Snyder-Cappione, J. E. (2019). Automated optimized parameters for T-distributed stochastic neighbor embedding improve visualization and analysis of large datasets. *Nature Communications*, *10*(1). https://doi.org/10.1038/s41467-019-13055-y

Breiman, L. (2001). Random Forests. *Machine Learning*.

Fix, E., & Hodges, J. L. (1989). *Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties*. *57*(3), 238–247.

Kobak, D., & Berens, P. (2019). The art of using t-SNE for single-cell transcriptomics. *Nature Communications*, *10*(1). https://doi.org/10.1038/s41467-019-13056-x

Pedregosa, F., Michel, V., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Vanderplas, J., Cournapeau, D., Varoquaux, G., Gramfort, A., Thirion, B., Dubourg, V., Passos, A., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830. http://scikit-learn.sourceforge.net.

van der Maaten, L., & Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, *9*, 2579–2605.