contrast\_brightness\_modify(*img, alpha, beta*)

Saturated contrast/brightness modification.

*g (i,j) =****α****.f (i,j) +****β***

With f(i,j) the original pixel value, **α** the desired contrast, **β** the desired brightness, g(i,j) the adjusted pixel value.

**Parameters:**

**img: *String like***

Direction/path to input image.

**alpha: *float, [1,2)***

Gain for each pixel intensity.

**beta*: float***

Offet for each pixel intensity.

**Returns:**

**new\_img*: ndarray***

Output image with modified contrast and brightness.

|  |
| --- |
|  |

automatic\_brightness\_and\_contrast(*image,clip\_hist\_percent=1,inverse\_color=False*)

Histogram equalization algorithm. This function clip a percentage of the histogram from the bottom.

**Parameters:**

**image: *array\_like***

Input image.

**clip\_hist\_percent: *float, Optional***

Clipped percentage of the maximum of grayscale histogram.

**inverse\_color*: bool, Optional***

Inverse the color of the image.

**Returns:**

**auto\_result*: ndarray***

Output image with modified contrast and brightness.

**alpha*: float***

Gain for each pixel intensity.

**beta*: float***

Offet for each pixel intensity.

|  |
| --- |
|  |

adjust\_gamma(*image,* *gamma=1.0*)

Gamma correction algorithm.

**Parameters:**

**image: *array\_like***

Input image.

**gamma*: float, optional***

gamma coefficient for the algorithm.

**Returns:**

**cv2.LUT(image, table)*: ndarray***

Output image with modified gamma coefficient.

|  |
| --- |
|  |

image\_treatment(*name\_img,inverse\_color=False,kernel\_morpho=5,open\_iter=1,close\_iter=1,clear\_bder=False*)

Image process algorithm.

**Parameters:**

**name\_img: *string like***

Direction to the input image.

**inverse\_color*: bool, optional***

Inverse the color of the image.

**kernel\_morpho*: int, optional***

Size of the kernel for morphological operations.

**open\_iter*: int, optional***

Number of iterations of morphological opening.

**close\_iter*: int, optional***

Number of iterations of morphological closing.

**clear\_bder*: bool, optional***

Clear object in contact with border.

**Returns:**

**binary*: ndarray***

Output binarized image.

**alpha*: float***

Gain for each pixel intensity.

**beta*: float***

Offset for each pixel intensity.

image\_treatment\_manuel(*name\_img,inverse\_color=False,kernel\_morpho=5,open\_iter=1,close\_iter=1,clear\_bder=False,alpha=1, beta=0*)

Image process algorithm with manual input of contrast and brightness.

**Parameters:**

**name\_img: *string like***

Direction to the input image.

**inverse\_color*: bool, optional***

Inverse the color of the image.

**kernel\_morpho*: int, optional***

Size of the kernel for morphological operations.

**open\_iter*: int, optional***

Number of iterations of morphological opening.

**close\_iter*: int, optional***

Number of iterations of morphological closing.

**clear\_bder*: bool, optional***

Clear object in contact with border.

**alpha*: float, optional***

Gain for each pixel intensity.

**beta*: float, optional***

Offset for each pixel intensity.

**Returns:**

**binary*: ndarray***

Output binarized image.

**alpha*: float***

Gain for each pixel intensity.

**beta*: float***

Offset for each pixel intensity.

|  |
| --- |
|  |

detect\_scale\_bar(*image\_path,physical\_length,inverse\_color= False*)

Detect and return the nm/pixel ratio of the scale bar.

**Parameters:**

**Image\_path: *string like***

Direction/path to the input image.

**physical\_length: *float, Optional***

The physical length in nanometer of the scale bar.

**inverse\_color*: bool, Optional***

Inverse the color of the image.

**Returns:**

**scale\_bar\_ratio*: float***

Ratio of the scale bar in nm/pixel.

|  |
| --- |
|  |

NP\_segmentation\_local\_max(*name\_img,min\_distance,* *dist\_max\_threshold=0.4,erode\_iter=1,open\_iter=0, kernel\_size=3*)

Watershed segmentation algorithm using local extremum.

**Parameters:**

**name\_img: *string like***

Direction/path to the input image.

**min\_distance: *float, Optional***

Minimum value of the pixel in the distance map to be considered as extremum.

**dist\_max\_threshold*: bool, Optional***

Percentage of the threshold in the distance map.

**erode\_iter*: int, Optional***

Number of iterations of morphological erosion.

**open\_iter*: bool, Optional***

Number of iterations of morphological opening.

**kernel\_size*: int, Optional***

Size of the kernel for morphological operations.

**Returns:**

**labels\_ws*: ndarray***

Segmented image as a 2D array.

NP\_segmentation\_fg\_bg(*name\_img,dist\_max\_threshold=0.4, erode\_iter=1,open\_iter=0,kernel\_size=3*)

Watershed segmentation algorithm using true background/foreground extraction.

**Parameters:**

**name\_img: *string like***

Direction/path to the input image.

**dist\_max\_threshold*: bool, Optional***

Percentage of the threshold in the distance map.

**erode\_iter*: int, Optional***

Number of iterations of morphological erosion.

**open\_iter*: bool, Optional***

Number of iterations of morphological opening.

**kernel\_size*: int, Optional***

Size of the kernel for morphological operations.

**Returns:**

**labels\_ws*: ndarray***

Segmented image as a 2D array.

|  |
| --- |
|  |

size\_histogram(*labels\_ws, pixel\_to\_nm, name\_img,bins = 100*)

Size histogram construction.

**Parameters:**

**labels\_ws: *ndarray***

Segmented image as a 2D array.

**pixel\_to\_nm*: float***

Pixel/nm ratio.

**name\_img*: string like***

Direction/path to the input image.

**bins*: int, Optional***

Number of intervals of the histogram.

**Returns:**

**radius*: np.array***

Array of size of segmented objects.

|  |
| --- |
|  |

divide\_histogram(*radius, edge\_radius*)

Divide the size histogram construction.

**Parameters:**

**radius*: np.array***

Input array of size of segmented objects.

**edge\_radius*: float***

Value to divide the array.

**Returns:**

**radius1*: np.array***

Array of size < edge\_radius of segmented objects.

**radius2*: np.array***

Array of size > edge\_radius of segmented objects.

|  |
| --- |
|  |

Gaussian\_fit (*radius, bins*)

Gaussian curve fit algorithm.

**Parameters:**

**radius*: np.array***

Input array of size of segmented objects.

**bins*: int***

Number of intervals of the histogram.

**Returns:**

**param\_optimised*: np.array***

Array of optimized values.

**param\_covariance\_matrix*: ndarray***

Covariance matrix of optimized values.

**x\_hist*: np.array***

Array of x value of the histogram.

**y\_hist*: np.array***

Array of y value of the histogram.

|  |
| --- |
|  |

plot\_Gaussian\_fit(*radius, bins*)

Plot the gaussian curve fit on the size histogram.

**Parameters:**

**radius*: np.array***

Input array of size of segmented objects.

**bins*: int***

Number of intervals of the histogram.

**Returns:**

**None**

|  |
| --- |
|  |

extract\_np(*i,img, labels\_ws,black\_bg\_color = False*)

Extract an object from the orignal image.

**Parameters:**

**i*: int***

Integer numerating the nanoparticle.

**img*: ndarray***

Original image.

**labels\_ws*: ndarray***

Segmented mask.

**black\_bg\_color*: bool, optional***

Indicate the color of the background.

**Returns:**

**None**

|  |
| --- |
|  |

extract\_binary\_np(*i,img, labels\_ws,black\_bg\_color = False*)

Extract an object from the orignal image in form of a binary mask.

**Parameters:**

**i*: int***

Integer numerating the nanoparticle.

**img*: ndarray***

Original image.

**labels\_ws*: ndarray***

Segmented mask.

**black\_bg\_color*: bool, optional***

Indicate the color of the background.

**Returns:**

**calibrated\_image: Image**

Image of the extracted object.

|  |
| --- |
|  |

testing\_image(*img,model,target\_size=(256,256), color\_mode='L'*)

Classify an image using a model.

**Parameters:**

**img*: ndarray***

Original image.

**model*: keras model***

Model used for classification.

**target\_size*: (int,int), optional***

Target size of the input image corresponding to the model input.

**color\_mode*: string, optional***

Color mode of the image (‘L’ for grayscale, ‘RGB’ for RGB).

**Returns:**

**result: np.array**

Output vector of probability.

**test\_image: ndarray**

Original image with expanded dimension for classification label.

|  |
| --- |
|  |

show\_xplique(*model,img,label,total\_label,alpha,method*)

AI explainable via Xplique. This Xplique function is only compatible for this whole process and notebook "Example of usage". In case of using Xplique for a specific image, please check out the examples of Xplique notebook.

**Parameters:**

**model*: keras model***

Model used for classification.

**img*: ndarray***

Original image.

**label*: int***

Classified label (usually np.argmax(result)).

**total\_label*: int***

Total number of classes.

**alpha*: float [0,1]***

Intensity of the explication image on original image.

**method*: string***

Explanation method (GradientInput, GradCAM, Saliency…).

**Returns:**

**None**

|  |
| --- |
|  |

Classification(*img\_name,model,total\_label,labels\_ws, target\_size=(256,256),color\_mode='L',black\_bg\_color = False*)

Classification plus showing extracted classes. Return a list of extracted mask of each class.

**Parameters:**

**Img\_name*: ndarray***

Original image.

**model*: keras model***

Model used for classification.

**total\_label*: int***

Total number of classes.

**labels\_ws*: ndarray***

Segmented mask.

**target\_size*: (int,int), optional***

Target size of the input image corresponding to the model input.

**color\_mode*: string, optional***

Color mode of the image (‘L’ for grayscale, ‘RGB’ for RGB).

**black\_bg\_color*: bool, optional***

Indicate the color of the background (corresponding to the model).

**Returns:**

**labels*: list***

List of extracted labels correspond to different classes.

|  |
| --- |
|  |