Main Functions Explained

## Table of Contents

[Table of Contents 1](#_Toc49416668)

[compute\_matches: 1](#_Toc49416669)

[load\_mask: 2](#_Toc49416670)

[score\_area: 2](#_Toc49416671)

[get\_IoU\_from\_matches 3](#_Toc49416672)

## compute\_matches:

Location:

“algo/mrcnn/utils”

Documentation:

"""

Finds matches between prediction and ground truth instances.

Returns:

gt\_match: 1-D array. For each GT box it has the index of the matched

predicted box.

pred\_match: 1-D array. For each predicted box, it has the index of

the matched ground truth box.

overlaps: [pred\_boxes, gt\_boxes] IoU overlaps.

"""

Deeper insight:

For each prediction compare with the all ground truth segments and looks for the first match. Match is the first ground truth segment for which the following holds:

1. The IoU between it to the prediction is higher than the given thresh
2. It has the same class label as the prediction
3. It hasn’t been assign to any previous prediction

## load\_mask:

Location:

“datautils/PDL1NetDataLoader”

Documentation:

"""

Generate instance masks for an image.

Returns:

masks: A bool array of shape [height, width, instance count] with

one mask per instance.

class\_ids: a 1D array of class IDs of the instance masks.

"""

Deeper insight:

Generates the masks for the ground truth data, using the assumption that the class ids order sets the classes priority in descending order, the higher the class id, the higher the priority.

For each annotation in the image creates Boolean image, and assign true to pixels in the polygon borders.

In order to remove intersections between the masks, and insure the model ground truth is deterministic, we followed the next algorithm. Combine all the masks from the same class to one image using logical or pixel-wise.

For each mask use bitwise and to remove intersections with the combined images of classes with higher priority.

The last part of the algorithm is to remove all the ‘other’ segments, because the network reads no class label pixels as background pixels. ‘other’ class meant letting the person that segments the data more flexibility, we added to the classes class other that in fact is exactly as background, but gave it the highest priority. That way, the person that make the segmentation labor, can segment large areas, and using the ‘other’ label it can define regions in the large segment that are background.

## score\_area:

Location:

algo\mrcnn\visualise\_pdl1

Documentation:

"""

calculate the area of the pdl1 positive out of all the cancerous cells,

score = num` pixels pdl1 \*pos\* / (num` pixels pdl1 \*pos\* + num` pixels pdl1 \*neg\*)

:param masks\_positive: ndarray [N, H, W] N number of positive segments

:param masks\_negative: ndarray [M, H, W] M number of negative segments

:return: score = num` pixels pdl1 \*pos\* / (num` pixels pdl1 \*pos\* + num` pixels pdl1 \*neg\*)

"""

Deeper insight:

Using the assumption that for a patch, cancer cells have the same mean area, we could find out the ratio of pd-l1 positive to negative, by counting the number of pixels in each segment and use simple ratio formula:

## get\_IoU\_from\_matches

Location:

algo\mrcnn\visualise\_pdl1

Documentation:

"""

if given an image, claculate the IoU of the segments in the image

:param match\_pred2gt: maps index of predicted segment to index of ground truth segment

:param matched\_classes: maps index of predicted segment to class number

:param ovelaps: maps [predicted segment index, gt segment index] to the IoU value of the segments

:return:

1. IoUs - IoU for all segments

2. IoUs\_classes - mean IoU per class

"""

Deeper insight:

This function returns 2 ways to compute the mean IoU. The first way, IoU is computed per slide, hence we get 4 IoU results for each class and then calculate the mean over all the segments. The second way to calculate the mean IoU, we first sum the IoU from all the segments (without taking into account the images that they came from) then we divide by the number of elements we summed.