The Whole Slide Image Segmentation Algorithm Performance Assessment Tool

(SegVal-WSI)

Arian Arab, Seyed Kahaki, Weijie Chen

Code Developer: Arian Arab (arian.arab@fda.hhs.gov)

Documentation and Testing: Arian Arab, Seyed Kahaki, Weijie Chen

Supervisor: Nicholas Petrick

Contents

Introduction

Applications of deep-learning (DL) algorithms in digital pathology have been in great interest in recent years [1]. For applications related to digital pathology (for example, in the case of designing applications for TILs-scoring, as will be discussed shortly), segmentation is often a major stage. For example, it is shown that tumor-infiltrating lymphocytes (TILs) are important in cancer prognosis [2]. In clinical practice, pathologists' visual assessment of TILs in biopsies and surgical resections of human epidermal growth factor receptor-2 positive (HER2+) and triple-negative breast cancers (TNBC) results in a quantitative score (TILs-score). Advances in ML algorithms in digital pathology pave the way for designing algorithms to automatically generate TILs-scores using whole slide images (WSIs). However, to design an automated TILs-scoring algorithm, the first stage is to design a segmentation algorithm to segment the tissue into tumoral and tumor-associated stromal regions. In later stages, a detection algorithm is also designed to detect "Lymphocytes" in the stromal regions. Based on the outputs of these segmentation and detection algorithms, a TILs-score, as a quantitative biomarker can be calculated automatically [3]. Hence, segmentation algorithms play a major role in a DL-based application in digital pathology.

With the segmentation annotations (in the form of segmentation masks) manually delineated by expert clinicians, a DL algorithm could be trained to automatically segment the tissue into tumoral and stromal regions. However, it is very burdensome and even infeasible to obtain segmentation annotations in the entire tissue regions of a WSI. This is due to the sheer size of WSIs in digital pathology applications in the range of 100,000 pixels in each of the two dimensions. Hence, a common practice is to select several regions of interests (ROIs) for segmentation annotations. As a result, segmentation annotations are obtained across several whole slide images, each containing several ROIs annotated in different locations.

One example is shown in the figure below wherein three ROIs in a single WSI are annotated by expert clinicians to train a segmentation algorithm.

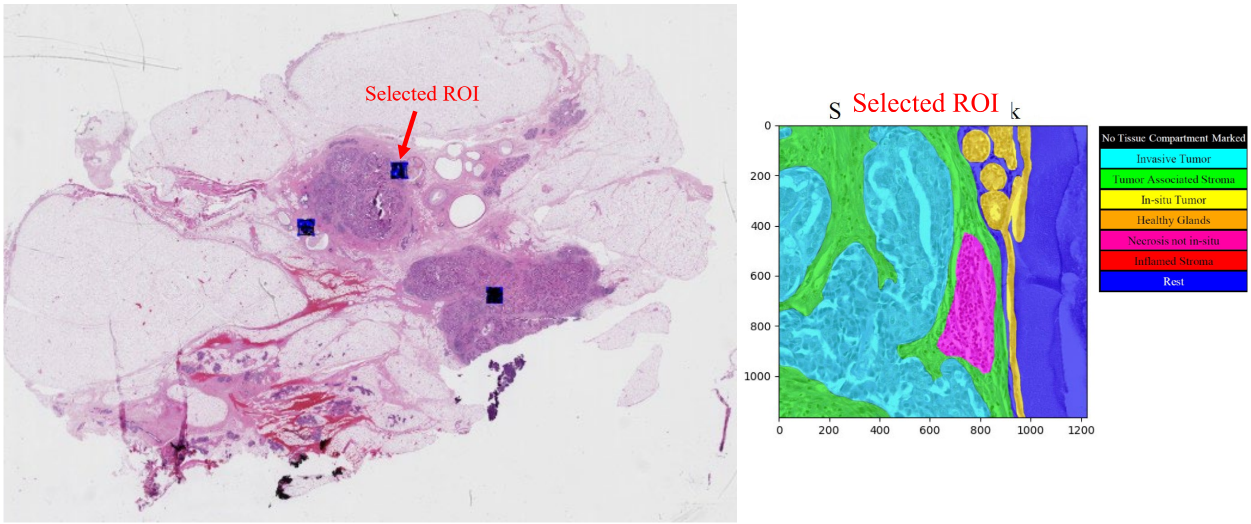


Figure 1. Left hand side shows the entire whole slide image usually with a size on the order of 100,000 by 100,000 pixels. Three selected ROIs are chosen and annotated. One ROI is shown on the right-hand side with the segmentation mask highlighting different tissue regions (image courtesy: https://tiger.grand-challenge.org).

After training a segmentation algorithm, these algorithms should be properly evaluated. As discussed previously, providing segmentation annotations in the entire tissue region of a whole slide image is not feasible. Instead, several ROIs are annotated. It is critically important to have proper test methodologies for aggregation when evaluating the performance of segmentation algorithms in digital pathology applications.

Note: We assume that the segmentation reference standard is already obtained through an established method in selected ROIs of a set of whole slide images and we only focus on evaluating the performance of the segmentation algorithms as compared to these reference standard annotations.

References

1) Janowczyk A, Madabhushi "A. Deep learning for digital pathology image analysis: A comprehensive tutorial with selected use cases". *J Pathol Inform*. 2016; 7:29. Jul 26, 2023.
2) C. Denkert et al., "Tumour-infiltrating lymphocytes and prognosis in different subtypes of breast cancer: a pooled analysis of 3771 patients treated with neoadjuvant therapy", The Lancet Oncology, 19, 2018, 40-50.
3) Arian Arab, et. al., "Effect of color normalization on deep learning segmentation models for tumor-infiltrating lymphocytes scoring using breast cancer histopathology images". Proc. SPIE 12471, Medical Imaging 2023: Digital and Computational Pathology, 124711H, April 6, 2023.

Method Description

We briefly describe the methods in this tool to evaluate performance of segmentation algorithms applied to digital pathology whole slide image applications.

We assume that the test dataset contains N WSIs from N patients with one WSI per patient (*i=1,2,* ...N, where *i* is the index of the i^{th} WSI and *N* is the total number of WSIs). Each WSI contains a number of ROIs annotated (M_1, M_2, M_3, … where M_i is the number of ROIs annotated for the i^{th} WSI, the total number of ROIs annotated across the entire set of WSIs is $M = \sum_{i=1}^{N} M_i$).

For each ROI, a reference segmentation mask is obtained. The reference segmentation masks will be compared to the algorithm-generated segmentation masks. If the segmentation algorithm is designed to segment the tissue into C classes, for each ROI, a $C \times C$ confusion matrix can be obtained, and a Dice score for each of the C class labels can be found (cm_{ij} refers to the confusion matrix of the j^{th} ROI of the i^{th} whole slide image).

Calculating Dice-Score from the Confusion Matrix

For a C-class segmentation task, by comparing the reference standard annotations to the algorithm's outputs, a confusion matrix of dimensions $C \times C$ can be obtained. From the confusion matrix, for each class, a Dice score can be calculated.

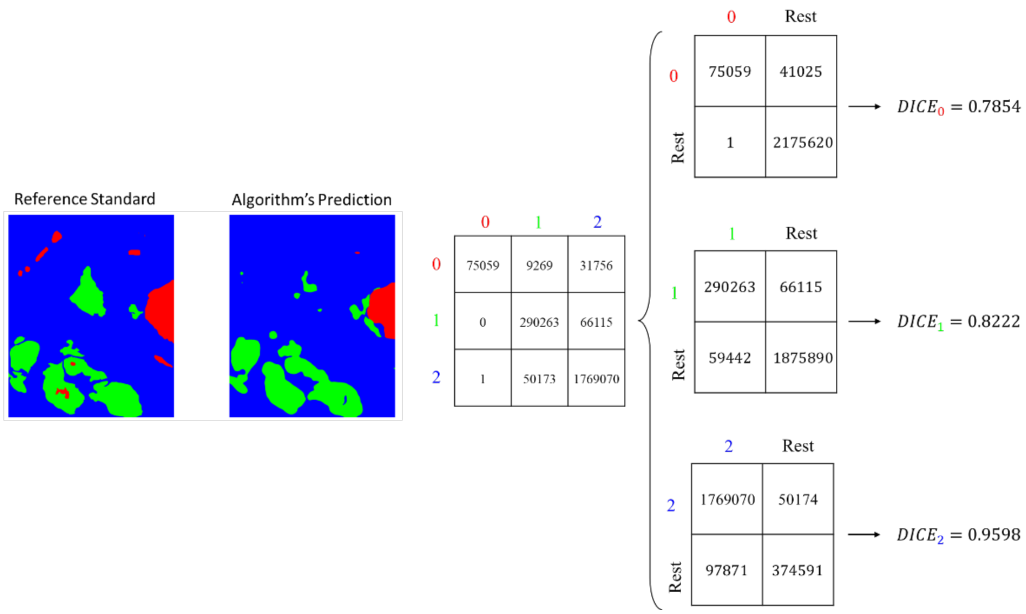
In a one-versus-the-rest approach, a class is defined as positive and the rest as negative, and a 2×2 confusion matrix can be obtained for each of the C classes. From the 2×2 confusion matrix, a Dice score for the positive class can be obtained (TP, FP, FN, and TN are obtained by comparing the algorithm's predictions and the ground truth labels at the pixel level):

	Predictions	
Ground Truth	TP	FN
	FP	TN

$$DICE = \frac{2TP}{2TP + FN + FP}$$

For each of the class labels, a Dice score can be obtained. If the reference standard annotations contain the positive class, a Dice score is obtained; otherwise, a Dice score is not assigned for that specific class (NaN).

For example, an ROI in a single WSI is shown below. The segmentation annotations contain three different class labels ($C = 3$, with label values of 0, 1, and 2). As a result, three Dice scores for each of the class labels are obtained:



	0	1	2
0	75059	9269	31756
1	0	290263	66115
2	1	50173	1769070

	0	Rest
0	75059	41025
Rest	1	2175620

→ $DICE_0 = 0.7854$

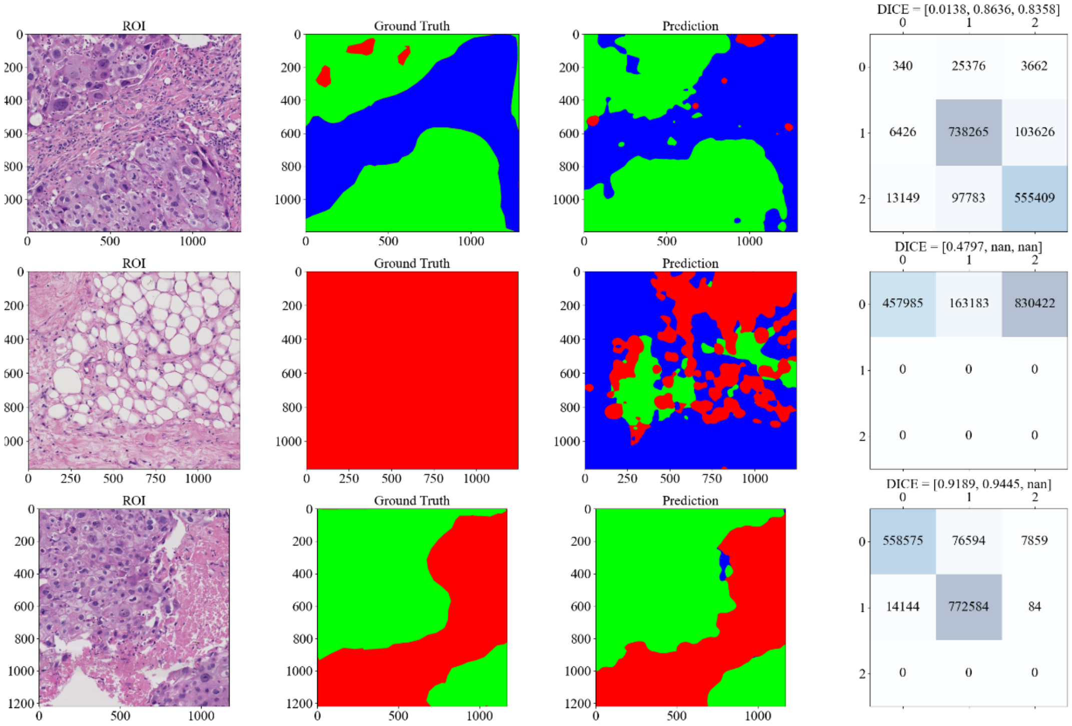
	1	Rest
1	290263	66115
Rest	59442	1875890

→ $DICE_1 = 0.8222$

	2	Rest
2	1769070	50174
Rest	97871	374591

→ $DICE_2 = 0.9598$

As another example, three ROIs in a single whole slide image, along with the reference standard annotations and the algorithm's predictions are shown in the figure below. As can be seen, for each of the ROIs, a confusion matrix can be obtained and since this is a 3-class segmentation task, for each of the class labels and for each of the ROIs a Dice score can be obtained:



DICE = [0.0138, 0.8636, 0.8358]

	0	1	2
0	340	25376	3662
1	6426	738265	103626
2	13149	97783	555409

DICE = [0.4797, nan, nan]

	0	1	2
0	457985	163183	830422
1	0	0	0
2	0	0	0

DICE = [0.9189, 0.9445, nan]

	0	1	2
0	558575	76594	7859
1	14144	772584	84
2	0	0	0

As can be seen from the figure above, for the first ROI, the reference standard annotations contain all the three class labels and hence three Dice scores are obtained for each of the three class labels (Dice scores of 0.0138, 0.8636, 0.8358).

For the second ROI, since two of the class labels are absent in the reference standard annotations, for these classes, Dice scores are not assigned ("NaN" values). A Dice score of 0.4797 is obtained for the class shown with the red color.

For the third ROI, since one of the class labels is absent in the reference standard annotation, a Dice score is not assigned for that class ("NaN" value) and Dice scores of 0.9189 and 0.9445 are obtained for the other two classes.

Calculating Dice-Score for the Entire Set of Whole Slide Images

We discuss three methods of obtaining a Dice score for the entire set of whole slide images. Here, we consider that each WSI contains different number of annotated ROIs (M_1, M_2, M_3, where M_i is the number of ROIs annotated for the i^{th} whole slide image) and N is the number of WSIs. The total number of ROIs is $M = \sum_{i=1}^{N} M_i$.

Method 1 (Pixel-Level aggregation): In the first method, we aggregate all the confusion matrices across all the ROIs and across all the WSIs to obtain one confusion matrix (CM, Equation 1). From this confusion matrix, CM, a Dice score can be obtained. This method weighs all the pixels across all the ROIs and across all the WSIs equally when aggregating all the confusion matrices (cm_{ij}) into a single confusion matrix (CM):

$$CM = \sum_{i=1}^{N}\sum_{j=1}^{M_i} cm_{ij} \rightarrow DICE = F_{DICE}(CM) \qquad (1)$$

Here, $F_{DICE}(CM)$ is the Dice scores obtained from the confusion matrix (CM).

Method 2 (ROI-Level Analysis): In the second method, we first calculate a Dice score for each of the ROIs ($Dice_{ij}$ is the Dice score of the j^{th} ROI of the i^{th} WSI obtained from the confusion matrix cm_{ij}). We then average all the Dice scores across all the ROIs in all the WSIs. This method weighs all the ROIs across all the WSIs equally, Equation 2:

$$DICE = \frac{1}{M}\sum_{i=1}^{N}\sum_{j=1}^{M_i} DICE_{ij} \quad | \quad DICE_{ij} \neq NA; \qquad M = \sum_{i=1}^{N} M_i \qquad (2)$$

Method 3 (Slide-Level Analysis): In the third method, we first calculate a Dice score for each WSI ($Dice_i$ is the Dice score of the i^{th} whole slide image). We then average all the Dice scores across all WSIs. This method weighs all the WSIs equally, Equation 3:

$$DICE = \frac{1}{N}\sum_{i=1}^{N} DICE_i \quad | \quad DICE_i \neq NA \qquad (3)$$

However, a Dice score for the whole slide image can be obtained in two different ways: firstly, by weighing all the pixels across an entire whole slide image equally, Equation (3a), and secondly, by weighing all the ROIs across an entire whole slide image equally, Equation (3b):

$$DICE_i = F_{DICE}\left(\sum_{j=1}^{Mi} cm_{ij}\right) \qquad (3a)$$

$$DICE_i = \frac{1}{Mj}\sum_{j=1}^{M_j} DICE_{ij} \quad | \quad DICE_{ij} \neq NA \qquad (3b)$$

Confidence Intervals

To calculate confidence intervals around the point estimates of the Dice metric, we use the bootstrap method. To obtain one bootstrap sample, we randomly sample N WSIs with replacement from the original pool of the N WSIs in the test dataset. Each time a WSI is sampled from the pool of the N slides, all the ROIs of that slide is also selected. As a result, the three methods discussed above could be used to obtain Dice scores for the bootstrap sample. We repeat this bootstrap sampling k times (e.g., k = 5000) to obtain k estimates for the Dice scores. From these Dice scores, lower and upper bounds of the confidence interval at different confidence levels could be obtained.

Installation

The software package has been tested with Python 3.#.#. Required packages include “NumPy”, “Matplotlib” and “tqdm” that can be installed as follows:

```
pip install numpy matplotlib tqdm
```

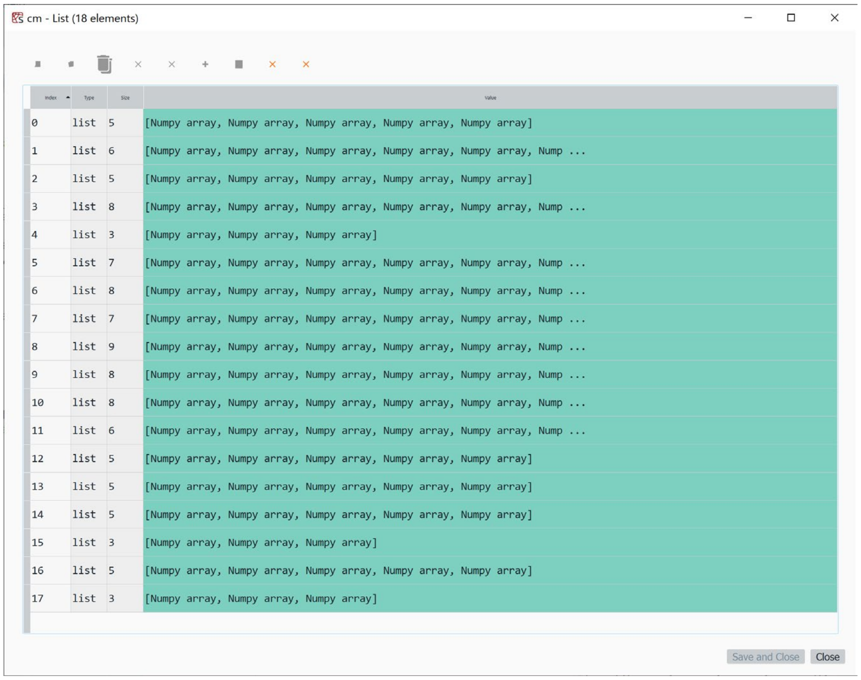
Usage

We describe each function in this package with an example test set of 18 WSIs, each containing from 3 to 9 ROIs, as provided in the following Numpy Array: “cm-example.npy”.

Confusion matrices for each of the ROIs of a whole slide image are obtained by comparing the reference standard annotation masks to the algorithm’s prediction masks. The data structure for the confusion matrices is a python *list* of WSIs wherein each element of the list is another list containing confusion matrices for all of the ROIs of that WSI.

Here, the segmentation task is a 3-class segmentation algorithm highlighting three different tissue regions.

```
import numpy as np
cm = list(np.load('cm-example.npy', allow_pickle = True))
```

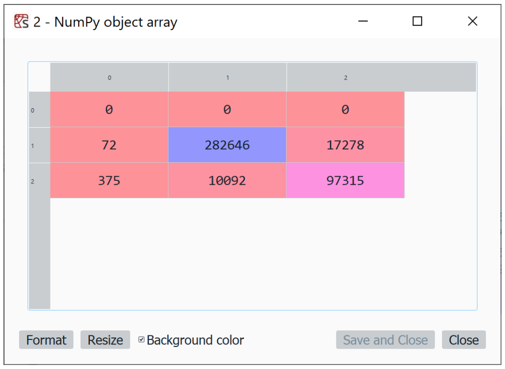


cm - List (18 elements)

Index	Type	Size	Value
0	list	5	[Numpy array, Numpy array, Numpy array, Numpy array, Numpy array]
1	list	6	[Numpy array, Numpy array, Numpy array, Numpy array, Numpy array, Nump ...
2	list	5	[Numpy array, Numpy array, Numpy array, Numpy array, Numpy array]
3	list	8	[Numpy array, Numpy array, Numpy array, Numpy array, Numpy array, Nump ...
4	list	3	[Numpy array, Numpy array, Numpy array]
5	list	7	[Numpy array, Numpy array, Numpy array, Numpy array, Numpy array, Nump ...
6	list	8	[Numpy array, Numpy array, Numpy array, Numpy array, Numpy array, Nump ...
7	list	7	[Numpy array, Numpy array, Numpy array, Numpy array, Numpy array, Nump ...
8	list	9	[Numpy array, Numpy array, Numpy array, Numpy array, Numpy array, Nump ...
9	list	8	[Numpy array, Numpy array, Numpy array, Numpy array, Numpy array, Nump ...
10	list	8	[Numpy array, Numpy array, Numpy array, Numpy array, Numpy array, Nump ...
11	list	6	[Numpy array, Numpy array, Numpy array, Numpy array, Numpy array, Nump ...
12	list	5	[Numpy array, Numpy array, Numpy array, Numpy array, Numpy array]
13	list	5	[Numpy array, Numpy array, Numpy array, Numpy array, Numpy array]
14	list	5	[Numpy array, Numpy array, Numpy array, Numpy array, Numpy array]
15	list	3	[Numpy array, Numpy array, Numpy array]
16	list	5	[Numpy array, Numpy array, Numpy array, Numpy array, Numpy array]
17	list	3	[Numpy array, Numpy array, Numpy array]

As can be seen from the figure above, “cm” is a list with 18 elements (18 WSIs), wherein each element (each WSI) contains a number of ROIs. For example, the first WSI contains 5 ROIs, the second WSI contains 6 ROIs and so on. For each ROI, a confusion matrix of size 3×3 is obtained.

For example, in the following, confusion matrix of the third ROI of the first WSI is given:



2 - NumPy object array

	0	1	2
0	0	0	0
1	72	282646	17278
2	375	10092	97315

In order to use this package, the “cm” variable (a list containing confusion matrices of all the ROIs of a WSI in a set of whole slide images) is used as its input. The input to this package should have the following format:

Input: A list wherein each element of this list is another list wherein each element is a NumPy array of size C×C (the NumPy array is the confusion matrix of an ROI) for a C-class segmentation task.

To obtain confusion matrices, you can use the following function: “**`calculate_cm.py`**”:

`calculate_cm(mask, pred, number_of_classes)`

Inputs:

- (NumPy Array) mask: mask annotations from the reference standard
- (NumPy Array) pred: algorithm’s prediction mask
- (int) number_of_classes: the number of classes in a C-class segmentation task

Output:

- (NumPy Array): confusion matrix of size C×C

As an example, a sample ROI with reference standard annotation masks and algorithm’s prediction masks are provided in the following NumPy array: “mask-pred-example.npy”. A 3×3 confusion matrix can be obtained by comparing the reference standard and algorithm’s prediction masks at the pixel level:

```
example_roi = np.load(‘mask-pred-example.npy’, allow_pickle = True).item(0)
img = example_roi[‘img’]
gt = example_roi[‘gt’]
pred = example_roi[‘prediction’]

from calculate_cm import calculate_cm
cm_roi = calculate_cm(gt, pred, number_of_classes=3)
```

“`cm_roi`”, is the confusion matrix obtained by comparing the ground truth (“gt”) and prediction masks (“pred”) at the pixel level, for a 3-class segmentation task.

To obtain the Dice score values using the three methods discussed previously, you can use the following function: “**`calculate_dice_wsi.py`**”:

`calculate_dice_wsi(cm)`

Input:

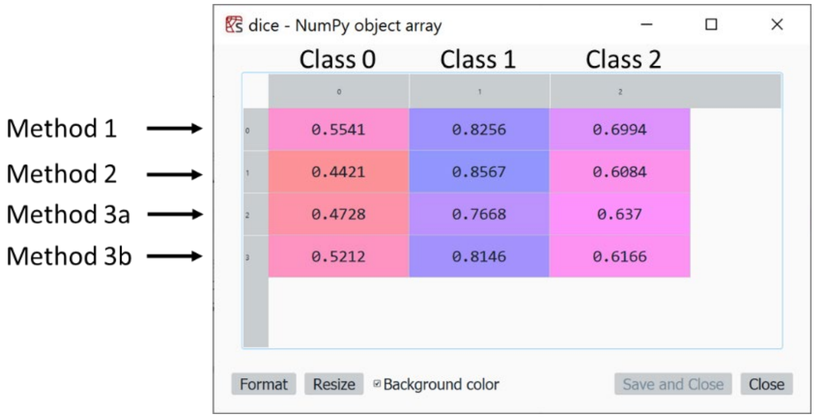
- (Python List) cm: A list wherein each element of this list is another list wherein each element is a NumPy array of size C×C (the NumPy array is the confusion matrix of an ROI) for a C-class segmentation task.

Output:

- (NumPy Array): final Dice scores for each of the class labels using three different methods. This NumPy array is of size (4×C) where in the 4 rows are the three different methods of calculating the final Dice scores and columns are the C-many class labels.

For example, for the set of 18 slides introduced above, the final Dice scores for each of the three class labels using the three different methods is obtained:

```
from calculate_dice_wsi import calculate_dice_wsi
dice = calculate_dice_wsi(cm)
```



	Class 0	Class 1	Class 2
Method 1	0.5541	0.8256	0.6994
Method 2	0.4421	0.8567	0.6084
Method 3a	0.4728	0.7668	0.637
Method 3b	0.5212	0.8146	0.6166

In order to find the confidence intervals around the point estimates you can use the following script: "**`calculate_dice_bootstrap_cases.py`**".

`calculate_dice_bootstrapce_cases(cm, nboots = 2000, confidence_level = 95)`

Inputs:

- (Python List) cm: A list wherein each element of this list is another list wherein each element is a NumPy array of size C×C (the NumPy array is the confusion matrix of an ROI) for a C-class segmentation task.
- (int) nboots: number of bootstrapped samples. Usually thousands of samples are chosen (default value = 2000).
- (int) confidence_level: The confidence level for reporting the lower and upper bounds of the confidence intervals.

Outputs:

- (NumPy Array): final Dice scores for each of the class labels using three different methods. This NumPy array is of size (4×C) where in the 4 rows are the three different methods of calculating the final Dice scores and columns are the C-many class labels.
- (NumPy Array): standard deviation of the final Dice scores obtained from the bootstrapped samples for each of the class labels using three different methods. This NumPy array is of size (4×C) where in the 4 rows are the three different methods of calculating the final Dice scores and columns are the C-many class labels.
- (NumPy Array): lower bound of the confidence interval of the final Dice scores obtained from the bootstrapped samples for each of the class labels using three different methods. This NumPy array is of size (4×C) where in the 4 rows are the three different methods of calculating the final Dice scores and columns are the C-many class labels.
- (NumPy Array): upper bound of the confidence interval of the final Dice scores obtained from the bootstrapped samples for each of the class labels using three different methods. This NumPy array is of size (4×C) where in the 4 rows are the three different methods of calculating the final Dice scores and columns are the C-many class labels.

In order to plot the point estimates along with the confidence intervals for each of the class labels as well as the three different methods to calculate the final Dice scores, the following script can be used: "**plot_dice_values.py**"

plot_dice_values(point_estimate, lb, ub)

Inputs:

- (NumPy Array) point estimates obtained from **calculate_dice_bootstrapce_cases**
- (NumPy Array) lower bound of the confidence intervals obtained from **calculate_dice_bootstrapce_cases**
- (NumPy Array) upper bound of the confidence intervals obtained from **calculate_dice_bootstrapce_cases**

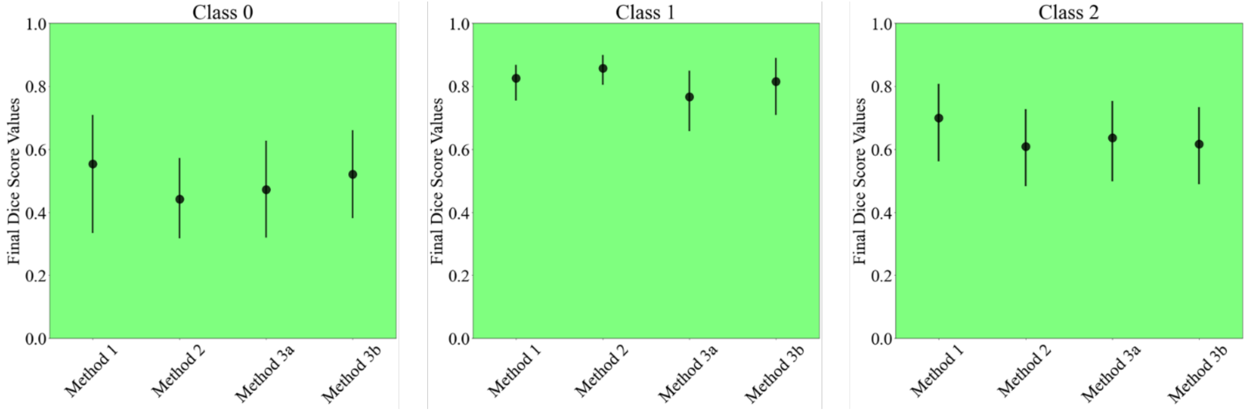
Outputs:

- (class "Figure"; Matplotlib library) Figures plotting the point estimates along with the confidence intervals for each of the class labels as well as the three different methods of calculating the final Dice scores.

For example, for the 18 whole slide images previously introduced, we can plot the final Dice scores along with the confidence intervals using 2000 bootstrapped samples at a confidence level of 95%:

```
from calculate_dice_bootstrap_cases import calculate_dice_bootstrap_cases
point_estimate,std,lb,ub = calculate_dice_bootstrap_cases(cm, nboots = 2000, confidence_level = 95)
```

```
from plot_dice_values import plot_dice_values
plot_dice_values(point_estimate, lb, ub)
```


Class 0
Class 1
Class 2
Final Dice Score Values
Method 1
Method 2
Method 3a
Method 3b