ANALYSIS OF THE MONUSAC 2020 CHALLENGE EVALUATION AND RESULTS: IMPLEMENTATION ERRORS

Adrien Foucart
Laboratory of Image Synthesis and Analysis
Université Libre de Bruxelles
Brussels, Belgium
adrien.foucart@ulb.be

Olivier Debeir
Laboratory of Image Synthesis and Analysis
Université Libre de Bruxelles
Brussels, Belgium

Christine Decaestecker
Laboratory of Image Synthesis and Analysis
Université Libre de Bruxelles
Brussels, Belgium

September 22, 2021

ABSTRACT

The MoNuSAC 2020 challenge was hosted at the ISBI 2020 conference, where the winners were announced. A subsequent publication was made in June 2021 in the IEEE Transactions on Medical Imaging. Challenge organizers, in addition to the leaderboard, released the evaluation code and visualisations of the prediction masks of the "top 5" teams. However, two key mistakes were made in the evaluation code, with the consequence of making the published leaderboard and the detailed per-organ and per-class results in the supplementary materials incorrect. We demonstrate the errors and their side-effect, and show without a doubt that the mistaken version of the code was indeed used to rank the algorithms in the challenge. Our results can be fully replicated with the code we provide on GitHub.

Keywords Digital pathology, challenge, nuclei segmentation, nuclei classification

1 Background

The MoNuSAC 2020 challenge was an official satellite event of ISBI 2020[1]. A pre-challenge preprint was published on ResearchGate[2], detailing the challenge procedure, the evaluation metric, and the expected submission format. Code for reading the annotations and for computing the evaluation metric was released on GitHub[3]. Examples of the submission file's format were also provided to the challenge participants via Google Drive[4]. Results of the challenge were announced during a workshop at ISBI 2020, with the workshop video made available on YouTube[5]. A post-challenge result summary and key findings analysis was published in the IEEE Transactions on Medical Imaging [Verma et al., 2021], with supplementary materials containing all the teams' submitted methods and more detailed metrics available on Google Drive[6]. Additionally, the "color-coded ground truth masks and predictions of top five teams" were released on the challenge website and made available alongside the challenge's training and test data.[7].

[1] Challenge website on grand-challenge.org
[2] Pre-publication preprint on ResearchGate
[3] Challenge code on GitHub
[4] Example submission on Google Drive
[5] Workshop video on YouTube
[6] Supplementary material on Google Drive
[7] Challenge data and top team predictions on grand-challenge.org

This shows a great transparency in the intentions of the organizers, providing a lot more information than many other digital pathology challenges. It provides us with a unique opportunity of being able to reproduce the results of a challenge, a key element in being able to trust the accuracy of those results, and of the insights and conclusions that came from those results.

Unfortunately, in the case of MoNuSAC 2020, a review of the available elements shows that key errors in the code resulted in erroneous results. It is clear that the mistakes in the published code were indeed used to rank the algorithms and declare the winners. The results of the challenge should therefore be re-computed and the subsequent publication revised.

All the code necessary to reproduce our results are available on GitHub: https://github.com/adfoucart/monusac-results-code-analysis

2 Overview of the dataset and predictions

The dataset contains images taken from whole-slide H&E stained tissue images. The training set contains images from 45 patients, and the test set from 25 patients. Tissues can come from four different organs (breast, kidney, lung and prostate). For each patient, regions ("sub-images") were extracted and their nuclei annotated. The annotations are stored as .xml files containing the vertices of polygons contouring the nuclei, and associated with one of four classes (Epithelial, Lymphocyte, Neutrophil, Macrophage). The provided datasets therefore contains, for each patient:

- SVS files containing the annotated regions of interest.
- A corresponding .xml file with the expert annotations.

The sub-images vary largely in their sizes and numbers of objects of interest, from tiny 100x100 pixels images with one or two nuclei, up to large 1500x1500 pixels images with hundreds of objects.

Participants were expected to provide, for each sub-image and for each class present in the image a separate .mat file containing the labelled objects of that class in that image (the "n-ary mask"). So if patient `PAT_1` has a sub-image `PAT_1_1` with the classes Epithelial and Neutrophil present, and a sub-image `PAT_1_2` with the classes Neutrophil and Macrophage, there should be 4 separate .mat files, each with labelled objects. The only requirement for those labels are that they are positive integers, with 0 being reserved for the background.

The provided predictions of the top teams, however, are color-coded as a single RGB image per sub-image, with each class being associated with a color, and borders being added to the objects to show the separation of close or overlapping nuclei. This means that, unfortunately, we do not have access to the raw .mat files with the actual labels and precise contours of each object that would allow us to fully reproduce the results of the challenge. As we will show, however, they are sufficient to demonstrate the problems in the computation of the metric.

3 Metric and described evaluation method

The metric used by the challenge is based on the "Panoptic Quality" (PQ), introduced in [Kirillov et al., 2019], and which essentially combines a segmentation score based on the average Intersection over Union of the true positives, and a detection score, which is simply the F1-score. For a given image, the PQ is computed per-class as:

$$PQ = \frac{\sum_{(p,g)\in TP} IoU(p,g)}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}$$

Where (p, g) are pairs of matching objects from the prediction and ground truth labelled masks. A "match" is defined as a pair of objects where $IoU > 0.5$.

The evaluation method described in [Verma et al., 2021] consists in computing the PQ per-class and per-image (PQ_c^i), then for each test image to compute the arithmetic mean of the class PQ ($PQ^i = \frac{1}{C}\sum_{c=1}^{C} PQ_c^i$), and to finally compute the global PQ as the average of the image PQs. The text of the publication implies that the "per-image" computation is done at the level of the *patient* (meaning that the IoUs, True Positives, False Positives and False Negatives are to be aggregated over all the "sub-images" of the patient before computing the PQ_c^i). This is clear from the sentences "Participants submitted a separate output file for each of the 25 test images" and "Arithmetic mean of the 25 PQ^i scores formed the final average panoptic quality", as there are 25 *patients* but more "sub-images" in the test set.

4 Description of the main error

The code for computing the PQ can be found in the "PQ_metric.ipynb" notebook on the challenge's GitHub[8]. The method Panoptic_quality(ground_truth_image, predicted_image) computes the PQ for one class of one image. It takes as input the ground truth "n-ary mask" (i.e. the mask of labelled objects) and the predicted n-ary mask. It first iterates through the ground truth objects:

```
for i in np.unique(ground_truth_image):
```

Then looks at all the intersecting predicted objects:

```
for j in np.unique(matched_image):
```

Then computes the IoU and, if a match is found (`if IOU> 0.5:`), the match and corresponding IoU are stored in a dictionary :

```
matched_instances[i] = j, IOU
```

With `i` the index of the ground truth object, `j` the index of the predicted object, and `IOU` the computed IoU of the matching pair.

This dictionary is then used to compute the TP, FP, FN and the avg IoUs of the TPs. This is done by creating a list of predicted indices present in the image :

```
pred_indx_list = np.unique(predicted_image)
pred_indx_list = np.array(pred_indx_list[1:])
```

With the second line removing the first index, which should always be 0 as `np.unique` returns an ordered list. This should only cause problem if a prediction map contains no background at all, which is admittedly a very unlikely scenario and isn't the main issue with the code.

The problem appears in the next step. The code iterates on the ground truth image's labels, and check if they are present in the matched instance dictionary. If that's the case, the index of the predicted object needs to be removed from the `pred_indx_list`, the TP count is incremented, and the IoU is added to a running sum. Otherwise, the FN count is incremented. Once all the ground truth objects have been checked, any remaining object in `pred_indx_list` should be a False Positive.

The errors appears when deleting the object from the `pred_indx_list`. The line in the code is:

```
pred_indx_list = np.delete(pred_indx_list, np.argwhere(pred_indx_list == [indx][0]))
```

When it should be:

```
pred_indx_list = np.delete(pred_indx_list, np.argwhere(pred_indx_list ==
matched_instances[indx][0]))
```

As with the former, `[indx][0]` will resolve to `indx`, which is the *ground truth object index*. If this particular ground truth label is not present in the predicted labels list, a True Positive will be correctly counted but a False Positive will be incorrectly added to the count, as the matching object will not have been removed from the `pred_indx_list`.

The resulting error will have no effect if and only if all the labels in `np.unique(ground_truth_image)` are present in `np.unique(predicted_image)` (as which particular label is removed from the `pred_indx_list` doesn't matter for the metric. The effect will be particularly strong, however, if the two lists are completely "unaligned". For instance, if the labels in the ground truth image are `[1,2,3,4]` and the labels in the predicted image are `[5,6,7,8]`, even if all the objects are perfectly matched, 4 False Positives will be counted by the provided method.

Unfortunately, this scenario is not unlikely at all, as there are two different strategies when generating the per-class prediction masks. As each class' prediction has to be sent separately, it makes sense to reset the label count for each class, so that labels for every prediction the "n-ary mask" will always have a set of objects starting at label 1. Conversely, as the goal of the challenge is to provide "segmentation and classification" for all classes in an image, it would also make sense to provide a unique label for each object in the image, and then to separate the objects by class. In that case, the labels could follow each other from class to class (for instance: Epithelial = `[1,2...N]`, Lymphocyte = `[N+1,N+2...M]`, etc., or they could be completely mixed.

Nothing in the provided instructions require one strategy over the others. The provided example .mat files illustrating the format chose the "labels following each other" method, so that the labels for a given class will typically only start at

[8] PQ metric notebook on GitHub

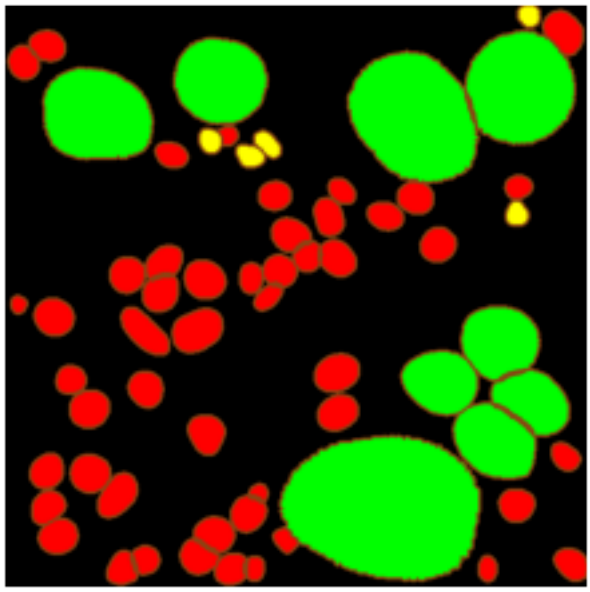

Figure 1: Color-coded predictions on the `TCGA-2Z-A9JG-01Z-00-DX1_6` image for the SJTU 426 team. Epithelial nuclei are in red, lymphocytes in yellow, neutrophils in blue, macrophages in green, and borders in brown.

1 for the first class present in the image. The code building the "n-ary masks" from the .xml annotations for the ground truth follow that same rule.

5 Demonstration of the main error

A quick test was first made (see supplementary material) to check that the code works according to our analysis. We constructed an artificial "ground truth" image containing three squares labelled `[1,2,3]`, and a "prediction" image with the same squares in the same positions, but labelled `[4,5,6]`. The IoUs of all matching pairs are therefore 1. A working code would therefore return a PQ of 1, as there are no false positive or false negative. Running the challenge's version of the code results in a PQ of 0.667, which is consistent with our understanding of the code: as the labels are completely unaligned, three FPs were mistakenly added to the count, leading to a result of $PQ = \frac{1}{3+0.5*3} = 0.667$. Our correction leads to the expected $PQ = 1$.

To test the impact of the bug, we use a single image from the "SJTU 426" team's predictions, which contains nuclei of multiple classes (see Figure 1). Using the .xml annotations for that image, and the code provided in the `n-ary_mask_generation.ipynb` file on GitHub[9], we can build the per-class n-ary masks. From the color-coded predictions, we can similarly extract all pixels of the class' color and re-label the objects to have the n-ary masks. The n-ary masks obtained with these methods for the "lymphocytes" class are shown in Figure 2.

Using these methods, the ground truth labels are `[48,49,50,51,51]` (continuing the count after the 47 epithelial nuclei), while the predicted labels are `[1,2,3,4,5]`. To illustrate the effect of the mistake, we can offset the predicted labels to increase the alignment and compute the PQ using the method provided by the challenge. We also compute the PQ with our corrected method (simply adding the missing `matched_instances`) to ensure that our correction solves the issue. Table 1 shows the results of this experiment. The PQ computed using the challenge's method vary from 0.385 to 0.501 depending on the alignment, while our corrected version's remains at 0.501 regardless of the value of the labels.

Was an incorrect score used by the challenge organizers? Based on the information that can be found on GitHub, yes. Looking at the history of the `PQ_metric.ipynb` file[10], we can see a "result dump" for several of the participating teams, showing the per-image, per-class PQ computed on the entire test set. According to this result dump, the score computed for that particular image for the SJTU 426 team was 0.381, which is very close to our "worst case scenario"

[9] N-ary mask generation code on GitHub.

[10] PQ metric file on March 20, 2020 (around the time of the publication of the leaderboard)

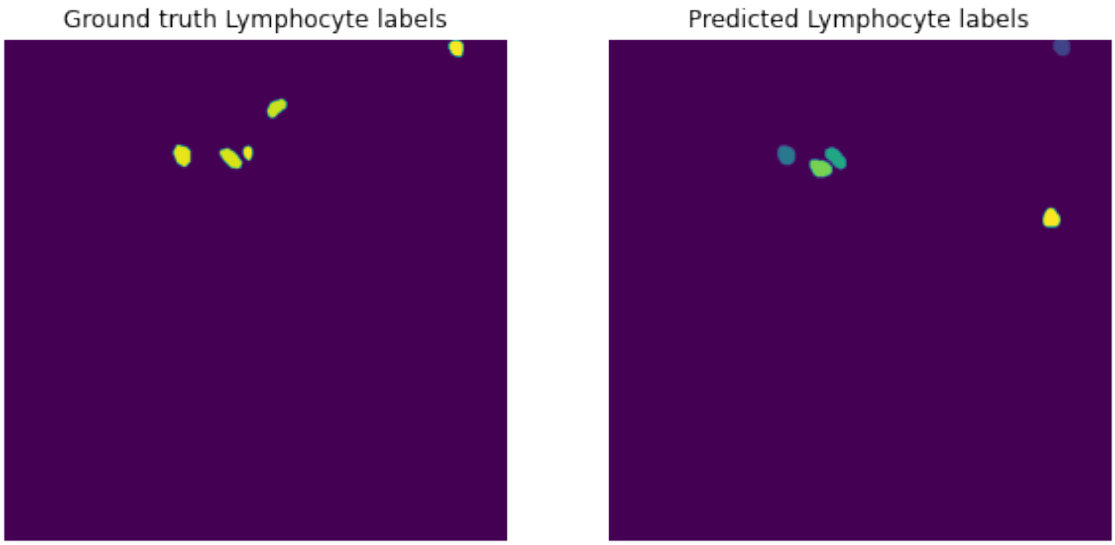

Figure 2: Ground truth and prediction labels for the Lymphocyte class. For the ground truth, the labels are generated from the .xml file using the code provided by the challenge. For the prediction mask, objects are extracted based on their color and relabelled using scikit-image's "label" method.

Table 1: Variation of the PQ with the "alignment" of the labels, as computed using the challenge method, and with our corrected version.

GT Labels	Prediction Labels	PQ (challenge)	PQ (corrected)
[48,49,50,51,52]	[1,2,3,4,5]	0.385	
[48,49,50,51,52]	[44,45,46,47,48]	0.385	
[48,49,50,51,52]	[45,46,47,48,49]	0.417	
[48,49,50,51,52]	[46,47,48,49,50]	0.455	
[48,49,50,51,52]	[47,48,49,50,51]	0.455	0.501
[48,49,50,51,52]	[48,49,50,51,52]	0.501	
[48,49,50,51,52]	[49,50,51,52,53]	0.501	
[48,49,50,51,52]	[50,51,52,53,54]	0.455	
[48,49,50,51,52]	[51,52,53,54,55]	0.417	

score of 0.385. The difference is likely to be due to our PQ being computed based on the color-coded predictions instead of the raw .mat files provided by the participants. Our objects are therefore slightly smaller than the objects that the challenge organizers used.

6 Error in the detailed per-organ results

Using that same "result dump", we can recompute the detailed table of results published in the supplementary materials of the post-challenge publication. The SJTU 426 is the "L2" team in the result table. Comparing the published per-organ, per-class PQs and the same metric recomputed from the result dump in Table 2 shows that they are identical, except that the Macrophage and Neutrophil classes are inverted. Looking at all the codes, it seems likely to be an error in reporting the results in the table, as throughout the code the order used for the classes puts neutrophils before macrophages, and the PQ metric file shows that they are also stored in that order in the .xls sheet in which the results of the challenge were put.

7 Problem with the metric's aggregation method

As mentioned in section 3, the post-challenge publication makes it clear that the PQ_c^i metric should be computed per-class, per-patient, and not per sub-image. This makes sense, as we don't want to give the same weight on the overall average PQ to a small image with two nuclei than to a large image with hundreds of nuclei.

Table 2: Per-organ, per-class results published in the supplementary materials of the post-challenge publication, and recomputed from the result dump.

Organ	Class	PQ (reported)	PQ (recomputed)
Breast	Epithelial	0.667	0.667
	Lymphocyte	0.608	0.608
	Macrophage	0.446	0.342
	Neutrophil	0.342	0.446
Kidney	Epithelial	0.583	0.583
	Lymphocyte	0.542	0.542
	Macrophage	0.441	0.465
	Neutrophil	0.465	0.441
Lung	Epithelial	0.614	0.614
	Lymphocyte	0.529	0.529
	Macrophage	0.566	0.621
	Neutrophil	0.621	0.566
Prostate	Epithelial	0.705	0.705
	Lymphocyte	0.542	0.542
	Macrophage	0.612	0.426
	Neutrophil	0.426	0.612

This is, however, not the method that allowed us to reproduce the results in Table 2. In the result dump, all the PQ provided were per-class, per-sub-image, and we simply averaged them per-organ in order to match the results of the challenge.

Similarly, we can recompute the overall result for the SJTU 426 team by computing the $PQ^i = \frac{1}{C}\sum PQ^i_c$ for each sub-image, and then averaging them over the entire test set. This gives us the expected 0.579 reported in the leaderboard. Nothing in the provided code performs the necessary operation of first compiling the IoUs, TPs, FPs and FNs per-patient *before* computing the PQ^i_c. This means that, even if we correct the PQ computation method, we would still have a discrepancy between the results and the published methodology.

8 Possible problem with undetected false positives

In the `PQ_metric.ipynb` file, we have in addition to the method that computes the PQ metric a piece of code that compiles the results per-class and per-(sub-)image into a .xls file. It's from this file that, presumably, the PQs where then averaged to get the final results shown in the leaderboard.

The way that the code process a team's predicted mask is to start from a list of files taken from *the ground truth directory*. Each of this file will contain the n-ary mask for one class on one sub-image. Iterating through this list, the code then finds a corresponding .mat file in the team's predictions directory, which had to follow the same structure (`PATIENT_ID/PATIENT_ID_SUBIMAGE_ID/Classname/xxx.mat`). The PQ is then computed based on these two n-ary masks, and added to the .xls worksheet.

There is, however, no code provided that checks if there were additional files in the *team predictions directory* that have no corresponding file in the *ground truth directory*. In other words: if a team found Lymphocytes in a sub-image that didn't contain any in the ground truth, it appears that these Lymphocytes would not be counted at all. Indeed, no PQ would be computed in that case for the Lymphocytes class, and the "False Positives" would go undetected. Without having access to the original, raw .mat files provided by the teams, it's impossible to confirm that this step was not undertaken, but it does seem likely given the evidence that these False Positives are missed by the challenge evaluation.

One way to confirm that possibility is to recompute the full PQ metric based on the colorcoded masks for the SJTU 426 team. Using the erroneous matching function, we can try to compute the PQ per-sub-image using two different strategies to compute the PQ^i: either to take the average of the PQ^i_c where there is at least one instance of an object of class c in the ground truth (which appears to be what the challenge has done), or where there is at least one instance of an object of class c either in the ground truth or in the predictions (which account for all false positives). The results in Table 3 show that we get results much closer to the officially published PQs when we include the mistake in the False Positives count. The difference in results here are again likely to be due to the fact that our results come from the color-coded predictions, and with labels which will be different than in the team's nary masks, which given the PQ computation bug will impact the results.

Table 3: Average PQ for the SJTU 426 Team based on a mistaken false positive count, on a correct false positive count, and in the reported challenge results.

SJTU 426	**Incorrect FP count**	**Correct FP count**	**Published results**
avg PQ	0.554	0.424	0.579

Based on our analysis, 439 nuclei detected by the SJTU 426 team with no corresponding ground truth object are not counted as False Positives. 66 submitted "nary-masks" are thus ignored because of a lack of corresponding ground truth mask, while the challenge only considers the 162 masks with ground truth annotations.

9 Recomputing the challenge results for the top teams

We can try to look at the effect of those errors on the challenge results by taking the available team predictions (4 out of the 5 "top teams", as two of the links unfortunately point towards the same team) and to recompute the PQ metrics based on a) our best guess as to the process used to produce the challenge results, as explained above, and b) our corrections.

These corrections are:

- Use the corrected matching function.
- Aggregate the IOUs, TP, FP and FN per-patient before computing the PQ^i
- Correctly account for the false positives in classes that are not present in the ground truth for a given image.

Both of these results can be compared to the published results of the challenge.

Table 4: Average PQ for the four teams with available colorcoded predictions, with the re-implemented method likely used by the challenge, just the "matching" correction, just the "ignoring FP" correction, just the "aggregate per-patient" correction, our modified method implementing all corrections, and the published results of the challenge.

Team PQ (rank)	**Re-implemented**	**Matching**	**IgnoredFP**	**PerPatient**	**All**	**Published**
Amirreza Mahbod	0.554 (2)	0.593 (3)	0.424 (1)	0.571 (3)	0.559 (2)	0.561 (2)
IIAI	0.580 (1)	0.617 (1)	0.424 (1)	0.581 (1)	0.545 (3)	0.549 (3)
SharifHooshPardaz	0.541 (4)	0.570 (4)	0.413 (4)	0.551 (4)	0.541 (4)	0.536 (4)
SJTU 426	0.554 (2)	0.594 (2)	0.424 (1)	0.578 (2)	0.560 (1)	0.579 (1)

The effect of the errors shown in Table 4 is relatively similar for all four teams: the computation of the PQ per-sub-image, and the matching method error tends to lower the score, while the forgotten false positives obviously increase the score. Taken together, those changes tend to counteract each other, and the final results after all the correction are very close to the published results... for those four teams. Indeed, there is clearly a selection bias at hand here, as teams that were more penalized by these errors are likely to have ended up well outside of the top 5 of the leaderboard.

It should also be noted once again that our re-computed results are made from n-ary masks "reconstructed" from the released colorcoded predictions. Dilating each object even by a disk of radius 1 to get an object size more similar to the one that was probably originally submitted by the teams significantly alter the results, as can be seen in Table 5. Some teams are helped by this dilation, while others actually see their score decreasing, likely depending on whether their original segmentation had a bias towards over- or under-estimating the size of the objects.

Table 5: Average PQ for the four teams with available colorcoded predictions, using our fully corrected method on the base reconstructed nary masks (border-removed), and on the nary masks with each object dilated by a disk of radius 1.

Team PQ (rank)	**Border-removed**	**Dilated**	**Published**
Amirreza Mahbod	0.559 (2)	0.573 (1)	0.561 (2)
IIAI	0.545 (3)	0.561 (2)	0.549 (3)
SharifHooshPardaz	0.541 (4)	0.504 (4)	0.536 (4)
SJTU 426	0.560 (1)	0.555 (3)	0.579 (1)

This "dilated" result from Table 5 is likely to be the closest to the actual results of the challenge, when taking into account all corrections. More accurate results cannot be computed without having access to the original nary masks sent by the participants. Unfortunately, the challenge organizers have thus far declined to work with us to do this re-computation, despite being shown evidence of the main error in the matching function.

10 Conclusions

We have conclusively shown that the results published in [Verma et al., 2021] and on the MoNuSAC 2020 leaderboard are based on an erroneous implementation of the PQ metric. More specifically, the results:

- Mistakenly detect False Positives when there are indices in the set of ground truth label which are not present in the set of prediction labels.
- Do not count False Positives when there was no annotated object of that class in the ground truth.
- Compute the per-class "PQ_c^i" metric, and the per-image PQ^i metric on each sub-image of the test set instead of first aggregating the IOUs, TP, FP and FN per-patient, as implied in the publication. This hugely penalizes teams which made a single mistake in very small images.

Our re-implementation and correction of the method shows that, for the four "top-5" teams available, the effect of those errors tend to cancel each other out so that the scores remain relatively similar when implementing all the corrections.

Consequently, it is clear that the current results as they stand are invalid, and that the leaderboard and publication should be temporarily removed while the organizing team recomputes the correct scores and reanalyzes the results.

Conflicts of interest

The authors of this publication declare no conflict of interest. None of the authors participated in the MoNuSAC challenge, or has any stakes in the results of the challenge themselves, other than in being able to use the insights that could be gathered from correct results.

References

Ruchika Verma et al. Monusac2020: A multi-organ nuclei segmentation and classification challenge. *IEEE Transactions on Medical Imaging*, pages 1–1, 2021. doi:10.1109/TMI.2021.3085712.

Alexander Kirillov et al. Panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.