ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2.
Available from http://www.i6doc.com/en/.

An agile machine learning project in pharma - developing a Mask R-CNN-based web application for bacterial colony counting

Tanguy Naets[1], Maarten Huijsmans[1], Laurent Sorber[1],
Paul Smyth[2], Gaël de Lannoy[2]

1- Radix.ai, Brussels, Belgium

2- GSK Vaccines, Rixensart, Belgium

Abstract. We present a web application to assist lab technicians with the counting of different types of bacteria colonies. We use a Mask R-CNN model trained and tuned specifically to detect the number of BVG+ (virulent) and BVG- (avirulent) colonies. We achieve a $mAP_{IoU=.5}$ of 94 %. With these encouraging results, we see opportunities to bring the benefits of improved accuracy and time saved to nearby problems and labs such as generalising to other bacteria types and viral foci counting.

1 Introduction

Counting and differentiating between different types of bacterial colonies is a key step in vaccine development,. It is a time-consuming and error-prone task that prevents biologists from performing more meaningful and fulfilling work. Several approaches have tried to avoid this manual counting using traditional computer vision algorithms by thresholding Petri dish images into binary masks of background and CFUs (Colony-Forming-Units), followed by post-processing techniques to segment each CFU [1, 2, 3, 4]. These approaches often suffer from a lack of multi-class CFU counting, poor performance on overlapping CFUs and miscounting of agglomerated colonies. These approaches also require careful calibration and good exposure conditions to properly exclude the background.

Recently deep learning approaches for CFU counting and other biomedical image tasks became popular with the advent of *image segmentation* algorithms, particularly thanks to the U-Net [5] architecture. For example [6] made use of a classical convolutional neural network instance segmentation algorithm to count CFUs, while [7] tackled CFUs counting with a U-Net architecture and hosted their model in the cloud to make it directly available for users taking pictures with mobile phones. [8] later trained a U-Net model to count colonies of two different bacteria classes, which to our knowledge is the first attempt at automation of multi-class CFU counting.

There are two main challenges faced when counting colonies using image segmentation algorithms : 1) *strongly imbalanced classes by nature* : most pixels are background, requiring carefully constructed loss functions in order to obtain satisfactory colony pixels' predictions ; 2)*overlapping CFUs* : separating CFUs during post-processing of the colonies masks is often achieved based on same-class-pixels' connectivity. Therefore, as soon as two colonies of the same class are connected by one or more pixels, these CFUs are misleadingly considered as a single colony. [8] attempts to address overlapping CFUs by labeling images with an additional boundary class, breaking the connectivity of overlapping colonies. Agglomerates of CFUs can then be separated with processing techniques. Unfortunately, this technique increases the data imbalance, as the newly introduced class of boundary pixels is even less common than the CFU classes and as soon as boundaries are not predicted perfectly, overlapping colonies are reconnected and one can no longer distinguish the single entities of an agglomerate.

This paper addresses the two challenges described above by tackling the problem with an *instance segmentation* architecture. The latest major breakthrough in the field of instance segmentation occurred in 2017 with the publication of Mask R-CNN [9].

We believe that instance segmentation algorithms should be immune to imbalanced classes and overlaying CFUs because they need not classify background pixels, nor do they need to rely on a boundary class to make the distinction between overlapping objects. In this work, we develop a model using the robust and efficient Matterport framework [10] and train on 101 GSK laboratory images of Petri dishes containing two types of bacterial colonies [8]. In addition to building a performant CFU differentiator and counter, we seek to build a modern web application so that our model can be effectively used in production by lab technicians to automate the counting as well as downstream related tasks.

2 Agile machine learning project

This project was conducted with end-users (lab technicians) in the loop from the start. Two-week sprints keep stakeholders informed and give them the opportunity to easily adjust the direction and priorities of the project, and end-users receive a working demo of the application immediately after the first sprint. This way, we could let the demo live and update it as sprints progressed so that stakeholders could continuously experiment with the application, providing valuable feedback for us so as to go as fast as possible from a prototype to an effective application. Below we describe the deep-learning aspects of the project.

We split the GSK dataset of Petri dish images gathered by [8] into train (65 %), validation (15 %) and test (20 %) sets. Several key parameters were adapted to fit the nature of our use case. One of those is the lengths of square anchor sides. Indeed, in the RPN (Region Proposal Network), generating objects bounding boxes proposals adapted to the sizes of the objects we seek to predict in an application is key to successful training. For instance, BVG objects are fairly small compared to the Petri dish images and compared to other object recognition tasks. Training was performed iteratively as follows[1] :

We first use a small, pre-trained backbone (ResNet-50) to speed-up the first training iterations, where only network head layers are trained since the backbone has already learned to recognize useful features on the COCO dataset [11].

[1]Training was achieved on a single AWS p3.2xlarge machine, equipped with an NVIDIA Tesla V100 GPU.

We then unfreeze and train the backbone layers, which leads to good results on training data but mediocre performance on some validation images, likely due to the locations of CFUs on these images having never been seen in the training set. We then augment the training set in various ways : stochastically rotating images from -180 to 180 °, scalings, translations, and additive and multiplicative noises. We also increase the number of (now randomly generated) images to 500 per epoch [2]. This leads to good model performance on the validation set. We then trade our ResNet-50 backbone for a 101 coco pre-trained one. A loss of performance is observed at first, as expected, but the model regains its previous performance level after only a few epochs and results slightly improve throughout the remaining epochs. We run a final training with more aggressive augmentations to make our model able to generalize to images that are very noisy, low contrast and taken from various angles and heights. This slightly improves the model's results even further up to a plateau on the validation set.

The left-hand side of Table 1 summarizes our best model's results using the average mAP (mean Average Precision) on IoU (Intersection over Union) thresholds from 50 to 95 %, the mAP at IoU 50 and 70 %[3] and MAPE (Mean Absolute Percentage Error) on BVG- (avirulent), BVG+ (virulent) and all BVG counts[4]. One can see that the mAP at a 0.5 IoU threshold is close to 100 % (94.1 %) on the test set. Also, the overall MAPE (Mean Absolute Percentage Error) on the test set demonstrates that on average, the total counts on a given image is off by less than 3 %.

Table 1: Benchmarks (in percentage) of our best model without post-processing (left) and with post-processing (right). The higher the mAP and the lower the MAPE, the better.

	No post-processing			**Post-processing**		
	train	val	test	train	val	test
mAP IoU=.50:.05:.95	58.3	51.9	50.7	58.2	51.0	50.6
mAP IoU=.5	97.1	97.6	94.1	96.8	96.3	93.8
mAP IoU=.75	63.5	47.5	50.5	63.2	46.8	50.6
MAPE BVG-	7.7	2.2	12.5	9.3	14.4	14.3
MAPE BVG+	3.1	2.1	5.7	2.8	1.9	4.8
MAPE Tot	2.0	1.4	2.6	1.6	2.5	2.3

We add several post-processing steps in order to further refine the quality of the results. First, we observe that the model is sometimes unsure about whether an object is a BVG+ or BVG- and will typically generate two bounding boxes and masks for both classes. In those cases, we remove the least likely object, and mark the most likely one as unsure (BVG+ or BVG-) so that users can be easily

[2]This gives a good trade-off between training speed and backups of the model's parameters.

[3]To benchmark the overall performance of instance segmentation models, as defined and used in [11] challenges.

[4]To assess the quality of model on the actual counting use case (the lower the better).

notified about that and validate or invalidate our guess. Second, CFUs on the border of Petri dishes are typically discarded by counting procedure rules. We implement this feature by spotting the Petri boundary (i.e., the corresponding ellipse), shrink it slightly and exclude predictions which do not intersect or are not included in this shrunk ellipse. Third, in rare cases, dust grains on a Petri dish can be confused with CFUs. As dust grains are typically smaller than the colonies, we exclude predicted objects whose area can be considered as an outlier based on Laplace distributions percentiles computed for each image.

We search the space of these post-processing features parameters in order to optimize model's performance on train and validation sets. We particularly focus on the total and especially the BVG+ MAPE as these are of utmost importance for the end-users. The right-hand side of Tab. 1 shows our best model's results with the following post-processing steps applied to exclude the predicted BVGs in the following cases : probability below 70 %, least likely of two objects of different classes overlapping significantly (i.e. $IoU \geqslant 70$ %), outside of the 98 % version of the guessed ellipse delimiting the Petri dish, and outside of 99 % confidence interval ([0.5 %, 99.5 %]) of the Laplace distribution computed on areas of BVGs on the current Petri dish image. Compared to our model without pre-processing (see left-hand side of Tab. 1, we can see performance gains mainly showing on the MAPE, decreasing from 2.6 % to 2.3 % and from 5.7 % to 4.8 % for the total and BVG+ counts on the test set, respectively. This is achieved at the minor cost of a slight decrease in mAP performance (94.1 % to 93.8 %), again on the test set. Tables 2 and 3 show the original and normalized confusion matrices on the test set, respectively.

As instance segmentation algorithms can predict non-existing objects or miss existing ones, an additional empty class is added. The intersection of these additional row and column (i.e. the last matrix element) always remains empty as non-predicted non-existing objects never occur. Furthermore, conversely to non-diagonal elements of standard classes, the *Nothingness* elements, i.e. *Missed* or *Invented* objects, should be as low as possible as one never seeks to miss/invent existing/non-existing objects. With this in mind, one can see in Tab. 3 very few colonies get misclassified with less than 2 and 7 % of actual BVG+ and BVG-, respectively. Also, one can note that actual BVG+ are less often confused with BVG- (0.5 %) than the opposite (4.8 %). Moreover, the model misses or invent twice as much BVG+ than BVG-, i.e. 3 to 6 and 2 to 4 (see Tab. 2). Because the proportion of BVG+ is more than twice bigger than the BVG- counterpart, this shows the model actually invents or misses less BVG+, relatively.

Table 2: Confusion matrix on test set

	BVG- predicted	BVG+ predicted	Missed
BVG- actual	78	4	2
BVG+ actual	2	395	4
Invented	3	6	.

Table 3: Normalized confusion matrix on test set (in %)

	BVG- predicted	BVG+ predicted	Missed
BVG- actual	92.9	4.8	2.4
BVG+ actual	0.5	98.5	1.0
Invented	33.3	66.7	.

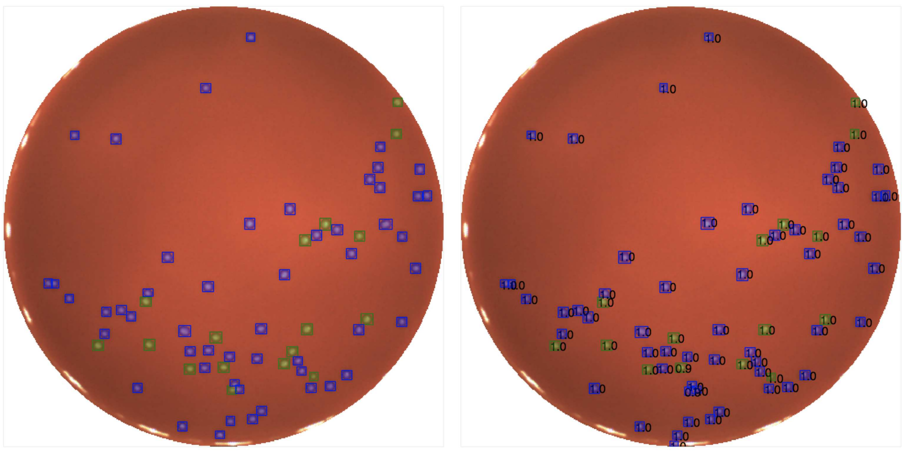

Fig. 1: Test data (left) and our corresponding best model's predictions (right)

Fig. 1 shows some examples of predicted CFUs on our test set with our best model. The overall visual inspection of the figure is promising. Most CFUs are spotted by the model and correctly classified. We can see most of the errors arise from CFUs on the image's border. One of our post-processing steps identifies the Petri dish border and only includes predictions within a 98 % reduced ellipse of the border. The reason not to further reduce the ellipse lies in the fact that images are taken from, and slightly asymmetrically cropped by, the aCOLyte[5]. This results in some images not completely encompassing some borders of the Petri dish. For instance, this is particularly flagrant on the bottom and left borders of the images presented in Fig. 1. Hence, using a greatly reduced ellipse of the border would lead to miss legitimate BVGs on the top and right borders. As a short term solution, we will display a conservative boundary ellipse and let the users change this to accommodate the asymmetric cropping of the images.

3 Conclusion and future work

We presented a Mask R-CNN model tuned and trained to count BVG- and BVG+ CFUs. This approach solves the imbalanced class problem discussed

[5]aCOLyte (Synbiosis, Cambridge, UK)is a hardware often used in CFU experiments.

above and significantly alleviates the problem of overlapping CFUs. Data augmentation, in particular rotations, is key to getting good performance with very few training examples. Slight improvements in the predictions quality could be achieved with a little more and better labelled data, although current results are already satisfactory to the operators. Another add-on could be setting-up a user-friendly interface embedded in our app so users can annotate new datasets. This would in turn facilitate the training and generalization to other species of bacteria. Finally, the areas of colonies are already computed and could later on be used to better estimate the actual quantity of bacteria in dishes.

4 Disclosure

This work was sponsored by GlaxoSmithKline Biologicals SA. All authors were involved in drafting the manuscript and approved the final version. The authors declare the following interest: GDL and PS are employees of the GSK group of companies and report ownership of GSK shares and/or restricted GSK shares. TN, LS, MH are employees of Radix.ai.

References

[1] Paul R Barber, Borivoj Vojnovic, Jane Kelly, Catherine R Mayes, Peter Boulton, Michael Woodcock, and Michael C Joiner. Automated counting of mammalian cell colonies. *Physics in Medicine & Biology*, 46(1):63, 2001.

[2] Maximilian Niyazi, Ismat Niyazi, and Claus Belka. Counting colonies of clonogenic assays by using densitometric software. *Radiation oncology*, 2(1):4, 2007.

[3] Silvio D Brugger, Christian Baumberger, Marcel Jost, Werner Jenni, Urs Brugger, and Kathrin Mühlemann. Automated counting of bacterial colony forming units on agar plates. *PloS one*, 7(3):e33695, 2012.

[4] Angelo Torelli, Ivo Wolf, Norbert Gretz, et al. Autocellseg: robust automatic colony forming unit (cfu)/cell analysis using adaptive image segmentation and easy-to-use post-editing techniques. *Scientific reports*, 8(1):7302, 2018.

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[6] Alessandro Ferrari, Stefano Lombardi, and Alberto Signoroni. Bacterial colony counting with convolutional neural networks in digital microbiology imaging. *Pattern Recognition*, 61:629–640, 2017.

[7] Hong-Ye Lin, Szu-Yu Chen, Chia-Lin Chung, and Yan-Fu Kuo. Counting bacterial colony on agar plates using deep convolutional neural network. In *2019 ASABE Annual International Meeting*, page 1. American Society of Agricultural and Biological Engineers, 2019.

[8] Thomas Beznik. Deep learning approach to bacterial colony detection. Master's thesis, UniversitÃ© catholique de Louvain, Belgium, 2019.

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[10] Waleed Abdulla. Mask r-cnn for object detection and instance segmentation on keras and tensorflow. `https://github.com/matterport/Mask_RCNN`, 2017.

[11] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.