Research Article

Julia Jeremias* and Constantin Pape*

Stamped counting for biomedical images

https://doi.org/10.1515/mim-2024-0021
Received November 7, 2024; accepted February 20, 2025;
published online April 8, 2025

Abstract: Counting objects is an important task in biomedical image analysis, for example to count cells in microscopy or microbial colonies in laboratory applications. Similar to most image analysis tasks, modern approaches to counting rely on deep learning, predominantly by solving an object detection or instance segmentation task and deriving the count from it. Here, we revisit regression-based counting, a simpler approach to counting, introducing a new method called *STACC* that makes use of the object size to derive the density map used as regression target. In our experiments, it consistently performs better than state-of-the-art methods automated for cell and microbial counting. We also provide a user-friendly tool to apply it to counting problems in practice.

Keywords: deep learning; image analysis; automated counting; bacterial colony counting; cell counting

1 Introduction

Counting is an important task in biomedical image analysis. Its applications include cell counting in microscopy or microbial colony counting in photos of culture media. In these applications the count is used to quantify an experiment or assay. For example, colony counting is used to analyze the degree of microbe contamination by taking samples from the environment, growing them on a culture medium and then counting the number of colonies. Another application is counting the number of colonies that grow after filtration in order to validate filter quality. Currently, the gold standard for counting in the laboratory setting is manual assessment. This work is tedious and also prone to inaccuracies due to attention errors. Hence, automation of the counting task is highly desirable.

Early work on automated (colony) counting used classical image processing, e.g. OpenCFU [1]. These approaches have largely been replaced by deep learning-based methods, which generalize better and thus enable counting for a wider set of imaging conditions. Most deep-learning based methods use counting by detection or counting by segmentation. These approaches are based on identifying all objects in an image by solving an object detection or an instance segmentation problem and then computing the unique number of objects. Examples include object detection architectures such as Faster R-CNN [2] or YOLO [3], or instance segmentation methods such as CellPose [4] or StarDist [5]. The colony counting problem in particular is most commonly addressed through object detection, see for example Majchrowska et al. [6] or Whipp and Dong [7]; with recent work employing vision transformers [8].

Despite its popularity, counting by detection or segmentation has some drawbacks. It requires expensive data annotations, where each object is outlined by a bounding box (for object detection) or instance mask (for instance segmentation). However, for counting a simpler form of annotations would suffice, either providing a count per image or marking each object with a single point. Furthermore, object detection and instance segmentation rely on complex architectures or sophisticated post-processing. An alternative is regression-based counting [9]. This approach directly predicts a count rather than detecting individual objects. Often, this is achieved by regressing a density map and deriving the count as its sum as in Lempitsky and Zisserman [9]. It has been implemented in ilastik for counting in biomedical images [10] using feature-based machine learning. Later work has also implemented regression-based counting with deep learning [11], where a deep neural network predicts the density map. More recently, it has been extended to support example-based counting [12]. However, regression-based counting is currently not in wide use for biomedical images, due to inferior performance compared to detection-based approaches.

Here, we revisit regression-based counting for biomedical images and show that it can yield results on-par or

***Corresponding authors: Julia Jeremias**, Sartorius AG, Göttingen, Germany, E-mail: julia.jeremias@Sartorius.com; and **Constantin Pape**, Institute for Computer Science, Georg-August-Universität Göttingen, Göttingen, Germany; and Cluster of Excellence 'Multiscale Bioimaging: from Molecular Machines to Networks of Excitable Cells' (MBExC), Georg-August-University Göttingen, Göttingen, Germany,
E-mail: constantin.pape@informatik.uni-goettingen.de

superior to detection-based methods. Our method uses a 2D U-Net [13] to regress a density map. We found that it is crucial to derive the target for this density map in a manner dependent on the object size. We compare different strategies for this step, including deriving local target densities from individual object sizes, an approach we call "Stamped Automated Colony Counting" (*STACC*). We perform experiments for two challenging counting problems, bacterial colony counting and cell counting. Furthermore, we implement a simple user-interface that enables life scientists to use our method. It is available at https://github.com/computational-cell-analytics/stacc. Overall, we present a simple yet highly effective method for automated counting, which is also depicted in Figure 1. It promises to automate time-consuming counting tasks in common analysis routines in the laboratory.

2 Materials and methods

We develop a new method for counting objects in biomedical images. It extends regression-based counting, which we review in Section 2.1, before introducing our method, *STACC*, in Section 2.2. We describe the datasets and metrics used for evaluating different counting methods in Sections 2.3 and 2.4.

2.1 Regression-based counting

Regression-based counting was introduced by Lempitsky and Zisserman [9]. They formulate counting as a regression problem, where the number of objects N_i in an image I_i is regressed with machine learning. More precisely, they regress a density map F_i, which has the same dimensions as I_i, that satisfies

$$\sum_j F_i^j = N_i \tag{1}$$

where j is the pixel index. The mapping of I_i to F_i is learned based on a training set with images and known object centers using feature-based machine learning.

This approach was extended to deep learning by Xie et al. [11], who use a convolutional neural network (CNN) to predict F_i. This method also requires a training set with images and known object centers. For each image, the density map F_i is created via

$$F_i = \tau\, G_{\sigma=2} * B_i. \tag{2}$$

Here, B_i is a binary image where each object center has the value 1 and all other pixels have the value 0, $G_{\sigma=2}$ is a Gaussian convolution operator with standard deviation $\sigma = 2$ and τ is a multiplicative factor that is set to 100. Note that F_i is correctly normalized by design, i.e., $\sum_j F_i^j =$

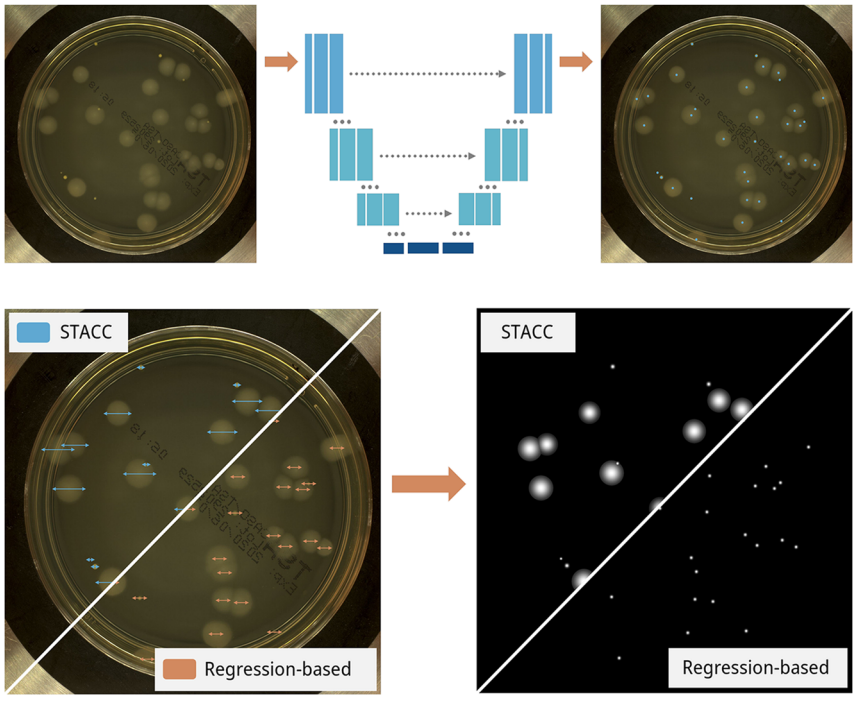

Figure 1: Overview of object counting with *STACC*: (Top) we predict a density map based on the input image with a 2D U-Net and find the object locations as maxima of this density map, shown here as blue dots on the image. (Bottom) *STACC* uses the per-object width to derive the target density map for U-Net training, unlike regular regression-based counting where a fixed width is used.

τN_i, up to numerical inaccuracies due to the discretization and truncation of the convolution. During training the CNN predicts an output map $\hat{F}_i$ that is compared to F_i with the L_2 loss. The parameters of the CNN are updated using stochastic gradient descent based on this loss. For a new image the object count can then be predicted by applying the CNN, summing over its output and dividing the result by τ.

2.2 STACC

The counting method described in the previous section is very simple. It can be trained based on data that contains object center annotations, requires only a few hyperparameters and can directly predict the count for new images without any post-processing. However, it performs worse in practice compared to detection-based methods, see for example our results in Table 1. Hence, detection-based methods have so far been more popular for counting. Here, we show that a modification of regression-based counting that takes into account the size of objects in order to construct the density map significantly improves counting results. We call our method "stamped automated colony counting", or *STACC*. The name derives from the idea of "stamping" the regression target with a size dependent kernel for each object location instead of using a uniform kernel for all objects.

Hence, *STACC* differs from regression-based counting according to Xie et al. [11] mainly in the construction of the density map. Instead of using a fixed standard deviation for the Gaussian convolutional operator, we derive a per-object standard deviation:

$$\sigma_k = \sqrt{\frac{-{s_k}^2}{2\log(\epsilon)}} \tag{3}$$

Here, s_k is the size of object k and $0 < \epsilon < 1$ is a truncation factor. In practice, we derive s_k from the width w_k of the bounding box of object k as $s_k = \frac{w_k - 1}{2}$. Given s_k for each object k in I_i, we compute F_i^{stacc} according to:

$$F_i^{\text{stacc}} = \max \left\{ \tau_k \, G_{\sigma=\sigma_k} * B_i^k \right\}_{k=1\ldots N_i} \tag{4}$$

Here, B_i^k denotes the binary map that is 1 at the center of object k and 0 otherwise. The factor τ_k is computed according to:

$$\tau_k = 8\,\pi\,{\sigma_k}^2 \tag{5}$$

We use the pixelwise max operation in Eq. (4) instead of a sum. This is done to avoid spurious maxima from overlapping Gaussians of adjacent objects that would result from summing up the individual object densities instead of using their maximum.

In order to solve the counting problem with *STACC*, we train a 2D U-Net [13] to predict F_i^{stacc} for the training set, using the L_2 loss to compare prediction and target. See Figure 1 for a graphical comparison of *STACC* and regular regression-based counting; see Section 3 for an overview of the training hyperparameters.

Note that F_i^{stacc} is not correctly normalized, i.e., its sum does not equal the number of obejcts in the image, due to the use of the maximum and the object dependent τ_k. In order to count the number of objects in a new image we thus compute the local maxima of the network predictions and count an object per maximum. This has the advantage of providing a prediction for object locations in addition to just the count. We use the local maximum implementation of scikit-image [14].

Note that *STACC* has a disadvantage compared to regular regression-based counting: it relies on an estimate for the object size s_k. Hence, the annotations required for training are more complex as the size (usually in the form of a bounding box) is needed in addition to the object location. Consequently, we investigate a different regression-based approach, where we use the median object size to derive the fixed standard deviation for F_i computed according to Eq. (2). In our experiments we derive this size as the median over all object sizes in the training set. In practice it could also be derived from first principles or estimated from a small sample of object sizes, enabling training with only object location annotations.

Table 1: Evaluation on the AGAR dataset for object detection methods, *STACC* and regression-based counting. All methods were trained for 100,000 iterations. Higher values correspond to a better quality for precision, recall and F1. For sMAPE and MAE lower scores correspond to better results.

Model	Precision	Recall	F1	sMAPE	MAE
Cascade R-CNN	0.89	0.88	0.89	8.41 %	4.56
Faster R-CNN	0.88	0.89	0.88	9.01 %	3.77
Regression-based $\sigma = 2$	0.88	0.65	0.70	27.23 %	10.23
Regression-based $\sigma = 22$	0.96	0.93	0.94	3.04 %	2.09
STACC	0.93	0.93	0.92	4.81 %	2.39

2.3 Datasets

We evaluate counting methods for two different tasks: microbial colony counting in photos of culture media and cell counting in phase-contrast microscopy. For the first task (microbial colony counting) we make use of two different datasets: the AGAR dataset [6], which contains images of bacterial colonies in AGAR medium, and an internal dataset

of microbial colonies growing on filter media. Originally, AGAR contains 18,000 images. The images are divided into three categories: countable, empty, and uncountable. The latter images have no colony labels due to their high colony count (over 300) and are therefore unsuited for training networks for the counting task. Without the uncountable images, 12,608 images remain. We adjusted these images to a uniform size of 2,928 × 2,928 pixels. Images of larger dimensions were center-cropped, images of smaller dimension were resized with a scaling factor, the annotations were updated accordingly. These changes allowed us to train on the entire image instead of patches, which provides the network with more spatial information. We make use of a train/test – split of 9,077 train and 2,522 test images. The internal filter dataset contains 1,145 images with bounding box annotations, which we split into 974 train and 171 test images.

For the second task (cell counting) we use the LIVECell dataset [15]. It contains 4,817 phase-contrast microscopy images of eight different cell lines with bounding box and mask annotations for each cell in the images. We use the splits introduced in the publication, which contain 3,253 train and 1,564 test images.

2.4 Metrics

We evaluate counting predictions using five different metrics. The systematic mean absolute percentage error (sMAPE) is defined as

$$\text{sMAPE} = \frac{100}{n} \sum_{i=1}^{n} \frac{|P_i - GT_i|}{|GT_i| + |P_i|}, \tag{6}$$

where P_i is the predicted count for image I_i, n the number of test images and GT_i the actual count for I_i. This metric gives a measure for the relative counting error. The mean absolute error (MAE) is defined as

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |P_i - \text{GT}_i|, \tag{7}$$

using the same definitions as for sMAPE. This metric gives a measure for the absolute counting error.

In addition, we compute precision p and recall r according to:

$$p = \frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i}, \quad r = \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i} \tag{8}$$

and the F_1-score as their harmonic mean: $F_1 = 2 * \frac{p*r}{p+r}$. For these metrics we compute the true positives (TP), false positives (FP), and false negatives (FN). The TPs are computed by matching predicted points and point annotations via linear cost assignment (hungarian matching). Here, the costs are derived from the distance of the predicted points to the point annotations – the closer to the true label, the lower the cost, such that the closest points will be matched. Based on this, FP and FN are calculated as $\text{FP}_i = P_i - \text{TP}_i$ and $\text{FN}_i = GT_i - \text{TP}_i$, where P_i is the predicted count and GT_i is the true count.

3 Results

We evaluate *STACC*, regression-based counting and detection/segmentation-based methods on the datasets described in Section 2.3. Unless specified otherwise, we use the following hyperparameters for network training:

- We use a 2D U-Net architecture with four levels, starting with an initial number of 32 features and doubling the number of features in each encoder level. We use convolutional filters with a kernel size of 3 and padding, max-pooling for downsampling in the encoder, linear interpolation for upsampling in the decoder and the same architecture as Ronneberger et al. [13] otherwise.
- We use PyTorch [16] and the torch-em library [17] to implement the network and training pipeline, using the ADAM [18] optimizer with an initial learning rate of 10^{-4} and PyTorch defaults otherwise.
- We divide the learning rate by a factor of 2 after the validation loss plateaus for 5 epochs.
- We use random horizontal flips and rotations by 90° as data augmentations.
- We train networks for 100,000 or 200,000 iterations, but then use the best checkpoint according to the validation loss for prediction.

All models were trained on an NVIDIA A100 GPU.

3.1 Colony counting

We first performed experiments on the AGAR dataset to compare *STACC* with regression-based counting and object detection. The first two methods are trained with our code, using the set-up described above. We train two object detection methods using the MMDetection framework [19]: Faster R-CNN [2] and Cascade R-CNN [20], which was the best method in the experiments performed in Majchrowska et al. [6]. The R-CNN architectures can be trained directly using the bounding box annotations provided by the dataset. For training *STACC* and regression-based methods we convert these annotations to center coordinates, corresponding to the bounding box centers, and to object sizes following Eq. (3).

Table 1 shows the results for these methods. We compare two regression-based approaches, $\sigma = 2$, which

corresponds to the settings of Xie et al. [11] and $\sigma = 22$, which corresponds to the median object size. We determine the best parameters for post-processing, corresponding to the minimal distance and absolute threshold of maxima for *STACC* and regression-based approaches and to the NMS threshold and maximum number of bounding boxes for the R-CNNs, on a separate validation set. We first note that the default setting of $\sigma = 2$ for regression-based counting yields inferior results compared to any other approach due to a low recall. Both *STACC* and regression-based counting with the median size perform better than the R-CNN architectures, with the median based approach yielding the best results.

Closer inspection of the *STACC* results indicated that this method performed not yet optimal for the given settings: the model did not yet converge in 100,000 training iterations, apparent by the fact that the best validation loss occured in the last epoch. Furthermore, visual inspection of the target density maps showed artifacts for very large and very small objects, likely due to numerical inaccuracies for Gaussian kernels with small or large σ values. Hence, we modified the procedure for computing the density map and clipped the per-object standard deviation values at the 22.5 % percentile (lower threshold) and 60 % percentile (upper threshold). The results for the updated experiments are shown in Table 2. Here, all methods were trained for 200,000 iterations (using the best checkpoint according to the validation loss for subsequent evaluation) and we compare two different *STACC* variants, with and without bounding the per-object standard deviation σ. We can see the advantage of *STACC* with bounds over the other approaches. Example predictions are shown in Figure 2. Here, we can see the qualitative differences between the methods: *STACC* and regression-based counting with the median yield quite similar results, but *STACC* performs better for images with heterogeneous size distributions and close-by colonies. This can be seen in the first row, where the arrows indicate close-by colonies that are predicted as a single colony by the median approach, but correctly predicted by *STACC*. The inferior prediction quality of regression-based counting with $\sigma = 2$ is immediately apparent as it misses many colonies.

Table 2: Evaluation on the AGAR dataset for *STACC* and regression-based methods. All methods were trained for 200,000 iterations.

Model	Precision	Recall	F1	sMAPE	MAE
Regression-based $\sigma = 2$	0.89	0.68	0.73	24.19 %	9.82
Regression-based $\sigma = 22$	0.96	0.94	0.95	2.68 %	2.03
STACC (no bounds)	0.94	0.94	0.93	3.82 %	1.85
STACC bounds	0.96	0.94	0.95	2.34 %	1.08

3.2 Transfer learning for colony counting

Next, we perform transfer learning experiments for the colony counting problem, to see how a network trained on

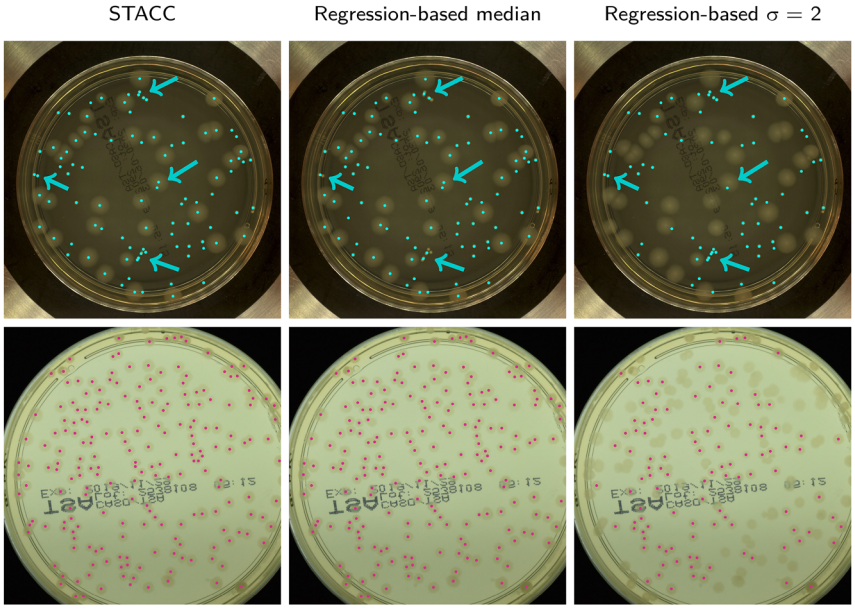

Figure 2: Qualitative results of different regression-based counting models on the AGAR dataset. Red and cyan overlays indicate predicted objects. Arrows indicate selected colonies that are correctly predicted by *STACC* but incorrectly predicted by the median-based approach or the fixed sigma approach.

a large annotated dataset, such as AGAR, can be applied to a smaller dataset, such as the filter data. We test three approaches, training networks only on the filter dataset (*scratch*), transfer learning from AGAR to the filter dataset (*filters*) and transfer learning using a combined dataset (*combined*). In the latter two cases, we either fine-tune exclusively on the filter dataset (*filters*), or on a combination of the filter and AGAR datasets (*combined*). In both cases, we initialize the networks with the weights of the respective best-performing model on AGAR. Note that for the regression-based median approach, the badwidth on the filter dataset is $\sigma = 8$; this model is initialized with the weights for the $\sigma = 22$ network from AGAR. Similarly, the bounds for *STACC* bounds are adapted for this dataset. Combining both datasets for finetuning aims to prevent catastrophic forgetting while enabling the analysis of bacterial growth on both agar plates and filter membranes. All networks are trained for 100,000 iterations.

The evaluation of these methods for the respective training data is shown in Table 3. Here, we again see that regression-based training with $\sigma = 2$ yields inferior results. Regression-based counting with the median size and *STACC* bounds yield similar results for all training datasets, showing that the filter dataset contains enough annotations to train from scratch. In addition, we evaluate the performance of these models on the AGAR dataset in Table 4. Here, we see that the performance of models that were finetuned exclusively on the filter dataset (*filters*) significantly decreases on AGAR (see also Table 2 for the pretrained model's performances), exhibiting catastrophic forgetting. However, models finetuned on the combined training data don't show this drop in performance. Overall, these results indicate the following best practices for fine-tuning networks on new datasets:

- If the new datasets is big enough, as is the case for the filter dataset, then training only on this data yields good results.
- If the performance on the original training data remains of interest, as is the case in practical applications, where a network should for example work for images of agar medium and filters, then finetuning on a combined dataset is the best option. Finetuning on only the new data would lead to catastrophic forgetting.

Figure 3 shows example predictions of the respective best-performing models from Table 3 on the filter dataset.

3.3 Cell counting

We then perform experiments for counting cells in phase-contrast microscopy images. We use the LIVECell dataset for these experiments, see also Section 2.3. We follow the

Table 3: Evaluation on the filter dataset for different counting methods trained on different training datasets. Models trained on the *scratch* training data were only trained on the filter dataset, models trained on *filters* were initialized with pretrained weights and then exclusively finetuned on the filter dataset, models trained on *combined* were also initialized with pretrained weights, and then finetuned on combined training data from the filter and AGAR datasets.

Model	Train data	Precision	Recall	F1	sMAPE	MAE
Regression-based $\sigma = 2$	scratch	0.96	0.90	0.93	5.94 %	5.71
Regression-based $\sigma = 2$	filters	0.96	0.86	0.90	8.87 %	10.86
Regression-based $\sigma = 2$	combined	0.99	0.92	0.95	4.31 %	6.22
Regression-based $\sigma = 8$	scratch	0.99	0.97	0.98	1.41 %	2.66
Regression-based $\sigma = 8$	filters	0.99	0.96	0.98	1.67 %	3.06
Regression-based $\sigma = 8$	combined	0.99	0.96	0.98	1.82 %	3.14
STACC bounds	scratch	0.98	0.96	0.97	1.96 %	2.34
STACC bounds	filters	0.99	0.96	0.98	1.73 %	2.92
STACC bounds	combined	0.97	0.96	0.96	2.37 %	3.22

Table 4: Evaluation of finetuned models from Table 3 on the AGAR dataset, showing catastrophic forgetting for models finetuned exclusively on *filters*.

Model	Train data	Precision	Recall	F1	sMAPE	MAE
Regression-based $\sigma = 2$	filters	0.51	0.51	0.46	37.45 %	12.78
Regression-based $\sigma = 2$	combined	0.86	0.61	0.67	31.26 %	11.13
Regression-based $\sigma = 8$	filters	0.56	0.70	0.58	31.75 %	9.26
Regression-based $\sigma = 8$	combined	0.96	0.93	0.94	3.27 %	1.45
STACC bounds	filters	0.51	0.64	0.50	40.48 %	18.18
STACC bounds	combined	0.95	0.93	0.94	3.27 %	1.29

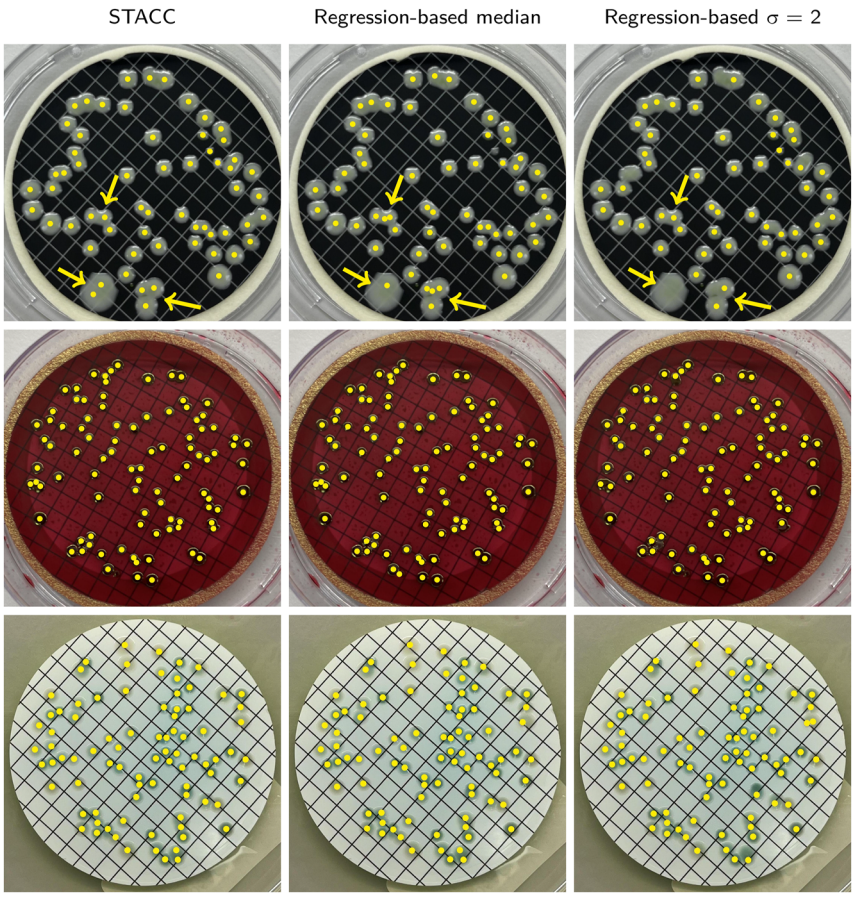

Figure 3: Qualitative results of the respective best model for the three different regression based approaches (STACC, median, fixed small sigma) on the filter dataset. Yellow overlays indicate predicted colonies. Arrows in the top row indicate selected colonies that were correctly predicted by STACC, but not by the median-based or fixed sigma approach.

same experimental set-up as before. For this dataset the median object size results in a standard deviation for the Gaussian of $\sigma = 4$. We also compare to CellPose [4], using the model trained on the LIVECell dataset provided by them. In order to evaluate instance segmentation results from CellPose with our metrics (see Section 2.4) we derive the centroid for each predicted mask.

The results for LIVECell are shown in Table 5. We can see a similar trend as for the other datasets: regression-based counting with $\sigma = 2$ performs badly, while the other regression-based approaches perform similarly well, with *STACC* bounds yielding the best results. The regression-based approaches (except $\sigma = 2$) all perform better than CellPose. Figure 4 shows example predictions on this dataset with the *STACC* bounds model.

Table 5: Evaluation on the LIVECell dataset.

Model	Precision	Recall	F1	sMAPE	MAE
Regression-based $\sigma = 2$	0.96	0.78	0.84	13.03 %	67.60
Regression-based $\sigma = 4$	0.94	0.90	0.91	5.91 %	39.80
STACC (no bounds)	0.93	0.90	0.91	6.01 %	42.33
STACC bounds	0.94	0.91	0.92	5.68 %	40.62
CellPose	0.90	0.86	0.87	7.31 %	62.65

3.4 Colony and cell type analysis

We also study the quality of results for the different types of bacterial colonies in AGAR and the different cell types in LIVECell. The corresponding results are shown for the 8 different cell types in Table 6 (LIVECell) and for the 5 different types of colonies in Table 7 (AGAR). Here, we use the STACC model with bounds trained on the combined AGAR and filter dataset for the AGAR result, and the LIVECell STACC model with bounds.

For AGAR, the challenges stem from the varied shapes and transparencies of both *Bacillus subtilis* and *Pseudomonas aeruginosa*, as shown in the second row of Figure 2. When these bacteria types grow closely together,

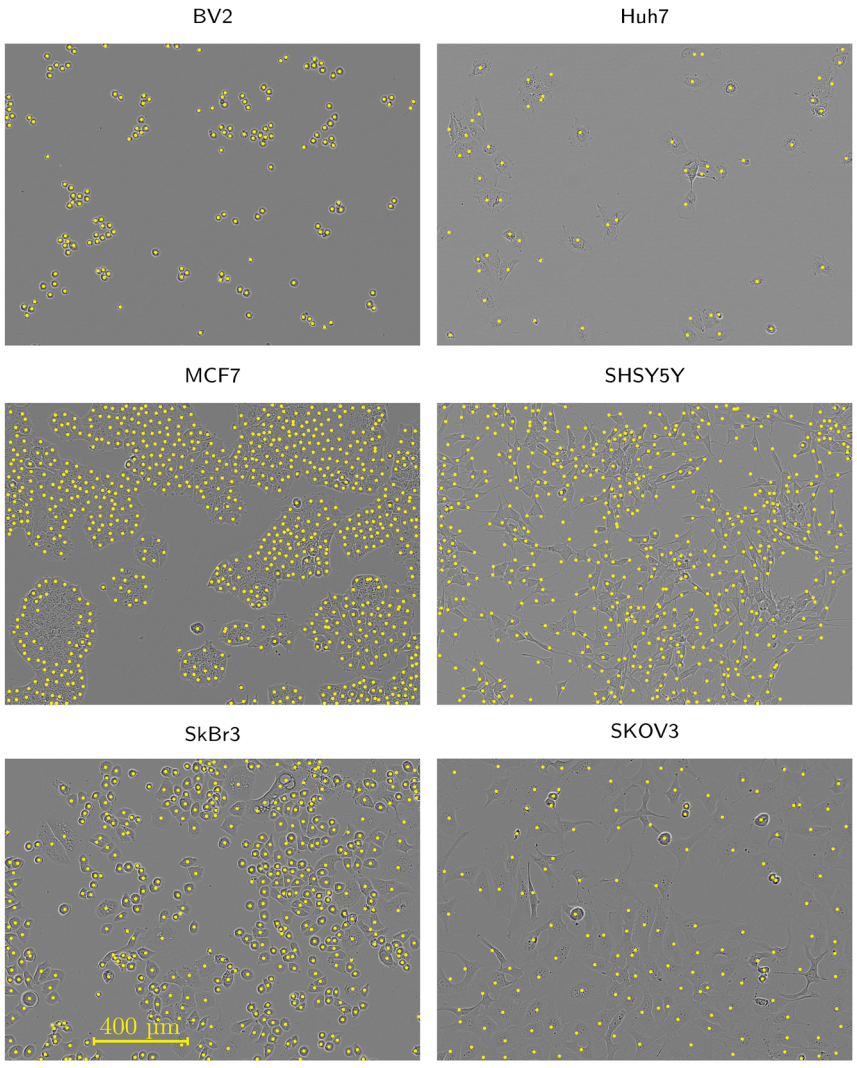

Figure 4: Predictions of the *STACC* bounds model on representative test images of different cell types from the LIVECell dataset. Yellow overlays indicate predicted cells. The scale bar is valid for all images.

Table 6: Mean performance values for each cell type.

Cell type	Precision	Recall	F1	sMAPE	MAE
A172	0.96	0.89	0.92	6.56 %	25.80
BT474	0.91	0.89	0.89	6.69 %	27.63
BV2	0.92	0.95	0.92	6.67 %	112.18
Huh7	0.88	0.88	0.87	6.42 %	7.79
MCF7	0.95	0.91	0.92	5.95 %	67.06
SHSY5Y	0.96	0.86	0.90	7.76 %	71.72
SkBr3	0.99	0.95	0.96	3.04 %	32.84
SKOV3	0.96	0.92	0.93	4.03 %	15.52

their colonies tend to merge, making it difficult to separate them accurately. In contrast, *Escherichia coli* and *Staphylococcus aureus* form more defined colonies, as seen in the first row of Figure 2. When *E. coli* colonies grow in close proximity, they establish a nearly distinct boundary between each other. For LIVECell the model performs best for cell types SkBr3 and SKOV3, these cells either have regular shapes (SkBr3) and/or do not show areas of very high confluence (SkBr3 and SKOV3). Conversely, the model performs worst for SHSY5Y, which has irregular shapes and

Table 7: Mean performance values for each bacteria type.

Bacteria type	Precision	Recall	F1	sMAPE	MAE
Bacillus subtilis	0.93	0.90	0.91	3.82 %	2.54
Candida albicans	0.93	0.94	0.93	2.96 %	1.34
Escherichia coli	0.97	0.96	0.96	1.81 %	0.62
Pseudomonas aeruginosa	0.95	0.91	0.92	5.32 %	1.69
Staphylococcus aureus	0.96	0.96	0.96	2.20 %	0.61

Figure 5: User interface of our napari plugin for automated counting. The plugin (right hand side) enables prediction with our models for colony or cell counting for images loaded in napari. It displays the predictions as the point layer "counting result", see points on the image, and also prints the number of predicted objects (lower right).

areas of high confluence with many overlapping cells. See also Figure 4 for exemplary images for six cell types.

3.5 User-friendly tool

As shown in our experiments, we provide a new method for object counting that is conceptually simple and that yields better results than existing methods. In order to make this method available to life scientists that face counting problems we build a user-friendly tool to apply it to new data.

Our tool is implemented as a napari plugin [21] and enables prediction in a graphical user interface, see Figure 5 for an overview. In addition to this tool our trained networks for colony and cell counting are available on BioImage.IO [22], a resource for sharing deep neural networks for bioimage analysis. This way they can be used within other image analysis software, such as Fiji via DeepImageJ [23].

Furthermore, we provide our training code in a well documented manner, so that users can train networks for other counting tasks. This includes functionality for fine-tuning of pretrained networks, including scripts that can leverage annotations from napari for easy training data generation.

4 Discussion

We have introduced a new method for counting in biomedical images. It extends regression-based counting by deriving the density map that is used as regression target in a manner depending on individual object sizes. We have demonstrated that this approach improves significantly over regular regression-based counting. In our experiments it also performs better than the state-of-the-art counting methods, which are based on object detection or instance segmentation, for two challenging problems. Furthermore, our experiments show two possible strategies for effective regression-based counting: our method, *STACC*, can be used if information about object position and size is available for each object in the training set. If the size information is not available, then regression-based counting using the median object size, which can be derived from first principles or estimated based on a small sample, can be used with only a small penalty in prediction quality.

A limitation of our method is that it does not provide semantic information. This information may be desired to count the number of different types of objects, such as the number of colonies of different bacterial stems or the number of different cell lines in co-cultures. Adding semantic information to the output is a possible extension for future work. Another possible extension is the combination of our method with Segment Anything [24] or a similar method for interactive segmentation. Here, the object positions predicted with our model could be used as input prompts to such a model in order to obtain an instance segmentation. Such a strategy is especially promising when combined with domain specific models, such as Segment Anything for Microscopy [25]. Finally, Spotiflow

Mantes et al. [26] is concurrent work that has explored spot detection for other bioimaging applications, for example spatial transcriptomics. This method shares conceptual similarities with our work, and could potentially also profit from integrating object size dependant target derivation.

Acknowledgments: We would like to express our gratitude to Sartorius AG for their support of this research through the Quantitative Cell Analytics Initiative (QuCellAI) and would also like to extend our thanks to all partners involved in the initiative for their contributions and valuable insights. The work of Constantin Pape was supported by Deutsche Forschungsgemeinschaft (DFG) under Germany's Excellence Strategy - EXC 2067/1-390729940. Special thanks to the laboratory technicians for their efforts in generating and annotating the filter dataset. We also gratefully acknowledge the computing time granted by the Resource Allocation Board and provided on the supercomputer Lise and Emmy at NHR@ZIB and NHR@Göttingen as part of the NHR infrastructure. The calculations for this research were conducted with computing resources under the project nim00007.

Research ethics: Not applicable.

Informed consent: Not applicable.

Author contributions: JJ and CP conceptualized the research and developed the methodology. JJ performed the experiments. JJ and CP implemented the software. JJ and CP prepared the manuscript.

Use of Large Language Models, AI and Machine Learning Tools: None declared.

Conflict of interest: JJ is an employe of Sartorius AG, who are developing commercial products for colony counting. CP is partially funded by Sartorius AG. J.

Research funding: None declared.

Data availability: The AGAR [6] and LIVECell [15] datasets are publicly available. The filter dataset will be made available upon request by the authors.

Patents: J and CP are currently filing a patent on automated colony counting. The patent will cover the core methodology submitted as part of this article. The patent rights will be held by Sartorius AG. Commercial use of the method described here will thus be subject to licensing.

Software availability: A test version of our software is available under the following link: https://owncloud.gwdg.de/index.php/s/2fJfD0TTYdcCNnj. Upon publication we will make all of our code, including the training code available under an open-source license. We will publish the two *STACC* models for colony and cell counting on BioImage.IO [22].

References

[1] Q. Geissmann, "Opencfu, a new free and open-source software to count cell colonies and other circular objects," *PLoS One*, vol. 8, no. 2, p. e54072, 2013.

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: towards real-time object detection with region proposal networks," *arXiv preprint arXiv:1506.01497*, 2015.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection," *arXiv preprint arXiv:1506.02640*, vol. 825, p. 2015, 2015.

[4] C. Stringer, T. Wang, M. Michaelos, and M. P. Cellpose, "A generalist algorithm for cellular segmentation," *Nat. Methods*, vol. 18, no. 1, pp. 100—106, 2021.

[5] U. Schmidt, M. Weigert, C. Broaddus, and G. Myers, "Cell detection with star-convex polygons," in *Medical Image Computing and Computer Assisted Intervention—MICCAI 2018: 21st International Conference, Granada, Spain, September 16—20, 2018, Proceedings, Part II 11*, Springer, 2018, pp. 265—273.

[6] S. Majchrowska, *et al*., "Agar a microbial colony dataset for deep learning detection," *arXiv preprint arXiv:2108.01234*, 2021.

[7] J. Whipp and A. Dong, "Yolo-based deep learning to automated bacterial colony counting," in *2022 IEEE Eighth International Conference on Multimedia Big Data (BigMM)*, IEEE, 2022, pp. 120—124.

[8] N. Ebert, D. Stricker, and O. Wasenmüller, "Transformer-based detection of microorganisms on high-resolution petri dish images," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 3961—3970.

[9] V. Lempitsky and A. Zisserman, "Learning to count objects in images," *Adv. Neural Inf. Process. Syst.*, vol. 23, 2010. Available at: https://papers.nips.cc/paper_files/paper/2010/hash/fe73f687e5bc5280214e0486b273a5f9-Abstract.html.

[10] L. Fiaschi, U. Köthe, R. Nair, and F. A. Hamprecht, "Learning to count with regression forest and structured labels," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, IEEE, 2012, pp. 2685—2688.

[11] W. Xie, J. A. Noble, and A. Zisserman, "Microscopy cell counting and detection with fully convolutional regression networks," *Comput. Methods Biomech. Biomed. Eng. Imag. Visual.*, vol. 6, no. 3, pp. 283—292, 2018.

[12] E. Lu, W. Xie, and A. Zisserman, "Class-agnostic counting," in *Computer Vision—ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2—6, 2018, Revised Selected Papers, Part III 14*, Springer, 2019, pp. 669—684.

[13] O. Ronneberger, P. Fischer, and T. Brox, "U-net: convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015: 18th International Conference, Munich, Germany, October2025-0002 5—9, 2015, Proceedings, Part III 18*, Springer, 2015, pp. 234—241.

[14] S. Van der Walt, *et al*., "scikit-image: image processing in python," *PeerJ*, vol. 2, p. e453, 2014. Available at: https://peerj.com/articles/453/.

[15] C. Edlund, *et al*., "Livecell — a large-scale dataset for label-free live cell segmentation," *Nat. Methods*, vol. 18, no. 9, pp. 1038—1045, 2021.

[16] A. Paszke, *et al*., "An imperative style, high-performance deep learning library," 2019. Available at: https://pytorch.org/docs/stable/generated/torch.nn.MSELoss.html#torch.nn.MSELoss.
[17] C. Pape, *et al*., "constantinpape/torch-em: 0.7.4," *Zenodo*, 2024, https://doi.org/10.5281/zenodo.13952442.
[18] D. P. Kingma, "Adam: a method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[19] K. Chen, *et al*., "Mmdetection: open mmlab detection toolbox and benchmark," *arXiv preprint arXiv:1906.07155*, 2019.
[20] N. Vasconcelos and Z. Cai, "Cascade r-cnn delving into high quality object detection," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017.
[21] N. Sofroniew, *et al*., "napari: a multi-dimensional image viewer for Python," 2024, https://doi.org/10.5281/zenodo.13863809.
[22] W. Ouyang, *et al*., "Bioimage model zoo: a community-driven resource for accessible deep learning in bioimage analysis," *bioRxiv*, pp. 2022—2106, 2022. Available at: https://www.biorxiv.org/content/10.1101/2022.06.07.495102v1.abstract.
[23] E. Gómez-de Mariscal, *et al*., "Deepimagej: a user-friendly environment to run deep learning models in imagej," *Nat. Methods*, vol. 18, no. 10, pp. 1192—1195, 2021.
[24] A. Kirillov, *et al*., "Segment anything," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4015—4026.
[25] A. Archit, *et al*., "Segment anything for microscopy," *Nat. Methods*, vol. 22, pp. 579—591, 2025.
[26] A. D. Mantes, *et al*., "Spotiflow: accurate and efficient spot detection for fluorescence microscopy with deep stereographic flow regression," *bioRxiv*, pp. 2024—2102, 2024, https://doi.org/10.1101/2024.02.01.578426.