# Detecting small objects and many of them: Using computer vision and deep learning to automate counting of collembola in petri dish samples

# Introduction

-Why do we need to count collembola in petri dishes

-What is the problem with the established method

-Developments in computer vision

-How can computer vision help solve our problem

-What are we presenting

Counting of collembola in petri dish samples in ecotoxicology experiments requires substantial manual labor by trained technicians. The process is time-consuming while the result is a simple census of the present individual collembola. Here, we investigated the feasibility of automating collembola counting based on a single image of a sample taken with a handheld camera and a deep learning object detection model.

Object detection models have gained significant attention for their ability to locate specific objects of interests in images. However, reliable detection of very small and very large numbers of objects remains a challenging task. In this regard, the present task is particularly challenging owing to the very small size of objects (down to a few pixels), low distinguishability of animals, large variation in number of objects in an image, complex background with structures that visually resemble animals, visual variation between samples, and variation in the conditions in which the image was taken, how the image was taken and the camera that was used.

We publish the code together with the trained model to facilitate others to implement automated analysis of ecotoxicology samples.

# Methods

* Overview of method
  + Standard method
  + Translating binary to bounding box format for model training
  + Splitting image/annotation data into training, validation, and testing sets
  + Slicing images/annotations for full resolution training
  + Training and testing neural network with sliding window approach

The method involved several steps: Translating binary filter images to bounding box format; splitting images into training, validation and testing sets; slicing images and associated annotations to a standard size in order to not loose information from large images because of downscaling; training and testing the neural network with a sliding window approach.

## 

## Generating training and testing data

Images had substantial variation in size, with the smallest image included being XxX and the largest XxX. We sliced to 640 x 640. Images were only sliced on dimensions larger than 640, i.e., an image with size 100x1280 would be sliced to two image of 100x640.

Training and testing object detection models requires annotated training and testing data. Specifically, images containing the objects of interest together with associated annotation data containing coordinates for bounding boxes delimiting the objects are required. Here, generation of annotation data was done through automated processing of existing outputs from human analysis. The human analysis results in a binary mask output of each petri dish image with black blobs delimiting animals. To convert these to bounding boxes, we applied automated contour detection and bounding box creation using Python OpenCV. The contour detection produces some errors when there are many objects very close to each other. Although the error rate seems low, for this test, images with a very high number of objects were discarded. Images from four tests were used (Test A-D). Only one image of each sample was used (i.e. for A1.1, A1.2, etc., only one image was included).

Contour detection and bounding box derivation was performed for a total of 127 images resulting in a total of 110.553 objects detected and annotated. See table x for distribution of images and objects between groups.

|  |  |  |
| --- | --- | --- |
| Group | Images | Objects |
| A | 44 | 66056 |
| B | 39 | 17479 |
| C | 16 | 17059 |
| D | 28 | 9959 |

|  |  |  |
| --- | --- | --- |
| TO DO | # images | # objects |
| Train | 89 |  |
| Val | 19 |  |
| Test | 19 |  |
| Total | 127 |  |

The images varied substantially in the number of annotated objects they contained. For each experimental group, we split the images into training, validation, and testing at a ratio of 70/15/15 at the level of images while optimizing for a ratio of 70/15/15 at the level of objects. If 70% of the images resulted in an unequal remainder or a decimal value, we rounded to a number that would ensure an equal remainder. This was done so that we could ensure an equal split between validation and testing in terms of number of images. We assigned images to groups by treating the problem as a linear programming problem with the number of images assigned to each group constrained to the 70/15/15 split while the distribution of images to groups were optimized towards lowest deviation from the 70/15/15 split at the level of objects assigned to each group.

## Deep learning methods

To test the applicability of object detection models for automatic detection and counting of collembola, we tested two model sizes of the object detection architecture YOLOv5: the small YOLOv5s which takes images with dimensions of 640x640 as input (hereafter referred to as YOLOv5 640), and the YOLOv5s6 which takes images of dimensions 1280x1280 (hereafter referred to as YOLOv5 1280).

Additionally, we tested the applicability of the Slicing Aided Hyper Inference (SAHI) method for each model. This method slices each image during inference in a sliding window manner and performs inference on each slice. The method is particularly relevant for detection of small objects.

Training length was set to 1000 epochs with early stopping with a patience of 100 epochs. Confidence threshold for prediction set to 0.1. Non-maximum suppression IOU threshold set to 0.45.

## Evaluating detection performance

Since YOLOv5+SAHI has a different output format than YOLOv5 on its own, customs scripts were created to 1) translate YOLOv5 output pickle files to standard YOLOv5 text files and 2) calculate detection performance variables (precision and recall) for both methods. Precision is the number of correct predictions out of all predictions made. Recall is the proportion of the true number of objects that were detected by the model. F1 is the weighted average of precision and recall. A value of 0.3 was used as the IOU threshold for whether bounding boxes overlapped sufficiently to be considered correct.

# Results

* Overview of original number of images, lowest resolution, highest resolution,
* Translating binary filters to bounding box format
  + lowest number of annotations, highest number of annotations
* Splitting images+annotations into training, validation, testing, with linear optimization (at image level but on condition of number of annotations in each set)
* Slicing images and annotations to avoid downscaling leverage full resolution information
  + Resulting number of images (mean number of annotations, sd?)
* Training and testing model
  + Prediction accuracy

The model trained on the 640x640 pixel images reached best validation accuracy at epoch 505. The model trained on the 1280x1280 pixel images reached best validation accuracy at epoch 705.

Table 1 Detection accuracy with the different models.

|  |  |  |  |
| --- | --- | --- | --- |
| Method (IOU 0.3) | Precision | Recall | F1 |
| YOLOv5 640 | 0.46 | 0.19 | 0.27 |
| YOLOv5 640 + SAHI (128) | 0.60 | 0.12 | 0.20 |
| YOLOv5 1280 | 0.54 | 0.56 | 0.55 |
| YOLOv5 1280 + SAHI (128) | 0.63 | 0.41 | 0.50 |

Table 2 Detection accuracy of YOLOv5 1280 + SAHI with different window sizes for the SAHI slicing (256, 128, 64).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model 1280** | SAHI 256 | | SAHI 128 | | SAHI 64 | |
| IOU 0.3 | Precision | Recall | Precision | Recall | Precision | Recall |
|  | 0.57 | 0.47 | 0.63 | 0.41 | 0.66 | 0.41 |

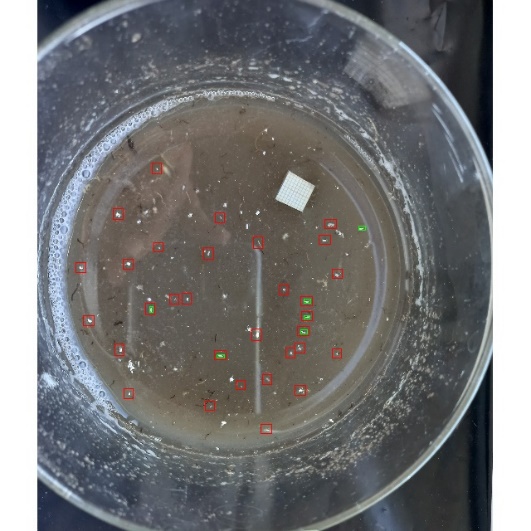


Fig. 1 Example of image with challenging background. Result from model 640 + SAHI 128. For this image, the precision was 0.16 and the recall 0.83.

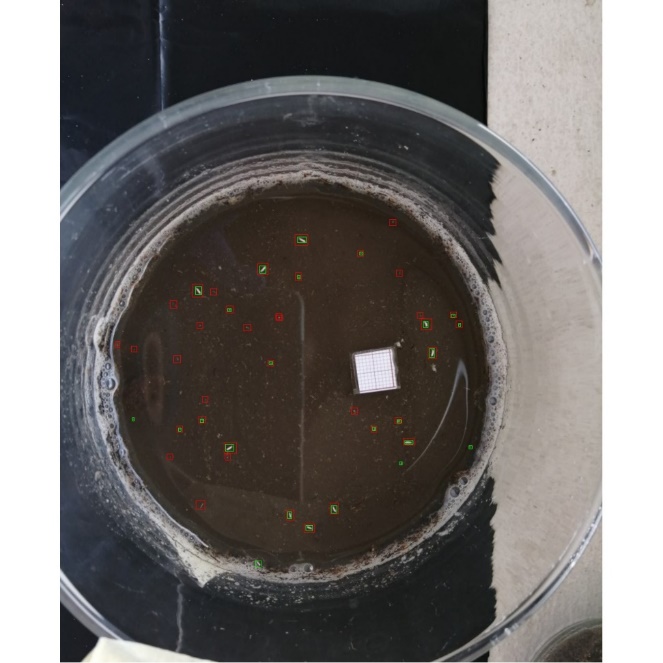


Fig. 2 Example image from test data set. Green bounding boxes are ground truth annotations derived from the binary images. Red boxes are detections made by the YOLOV5 1280 model + SAHI. In this case, most animals marked in the ground truth are detected by the model (high recall), but the model produces many false positives (low precision). Note the overlapping detections in one place (middle of image).

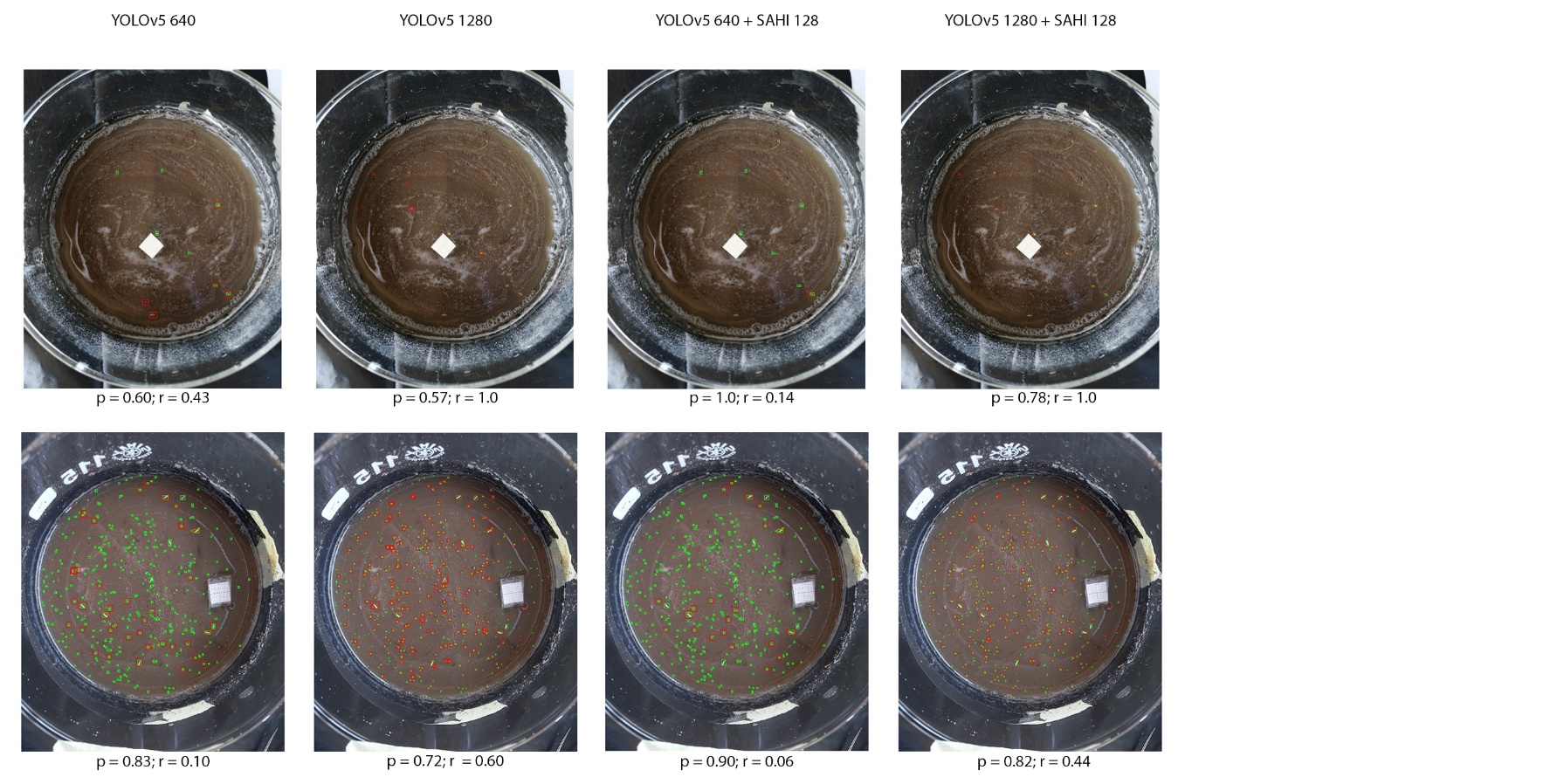


Fig. 3 Example outputs from the two different model sizes with and without application of SAHI (128x128 window size). P: precision, r: recall.

Table 3 Variation in performance between images (YOLOv5 1280 + SAHI 128).

|  |  |  |
| --- | --- | --- |
| Image | Precision | Recall |
| A11.1 | 0.78 | 1.0 |
| A17.1 | 0.54 | 0.79 |
| A31.1 | 0.42 | 0.71 |
| A38.1 | 0.73 | 0.50 |
| A49.1 | 0.10 | 1.0 |
| A6.1 | 0.60 | 0.78 |
| B37.1 | 0.53 | 0.39 |
| C3.1 | 0.48 | 0.85 |
| C7.3 | 0.62 | 0.68 |
| D23.1 | 0.82 | 0.44 |
| D4.1 | 0.20 | 0.20 |
| D9.1 | 0.66 | 0.09 |

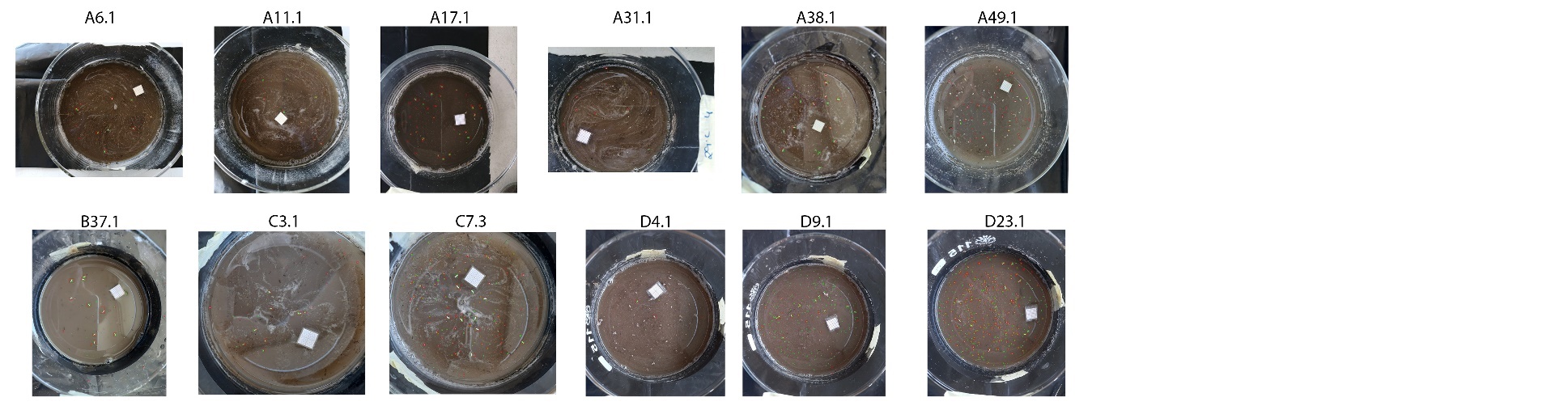


Fig. 4 Detection results for each test image for the YOLOv5 1280 + SAHI 128 model. See table 3 for accuracy metrics.

# Discussion

* What did we present

The improved precision with increased image dimensions and with the application of SAHI underlines that a main challenge of the task of detecting collembola is the very small object sizes relative to the full image.

## Overlapping detections

The models tended to produce overlapping detections. This will have some influence on the accuracy metrics, but it can be assumed to be minor, since the overlapping detections seems to be rare. Overlapping detections are usually removed by non-maximum suppression. However, for the present case with very small object, the NMS IOU threshold was likely set too high,

## Accuracy of the ground truth annotations

Considering the similarity between background and animals, there is likely some level of error in the annotations. On top of this, the automated extraction of bounding boxes from the binary images could introduce error, e.g. when there are many objects close to each other (all though in the present case, such images were discarded). If the accuracy of these ground truth annotations is in fact low, it will have detrimental effects on model accuracy. To investigate this issue it could be relevant to test the variation in manual annotations between annotators.

## Variation in accuracy between test images

The model accuracy generally varied substantially between images within the test set. This is likely because the model has not been trained on a training set of sufficient size and variation, which means the models fails to generalize between images.

Different issues with data means that the present experiment was carried out on a limited set of images.

Issues/challenges with the available data

* Contour errors when objects are many and close.
* Test E folder contains no original images (only b/w), so these cannot be used.
* Many original images without associated b/w image.
* Several versions of some b/w images. Used latest version in each case.
* Images have varied dimensions. Standardized by padding as described. This also means variation in size of objects (in pixels).