Instance Segmention on Large Scales Images

Sebastian Rassmann

October 1, 2021

1 Introduction

A popular tool for insect detection in collection box digitization efforts is *Inselect* [6]. It streamlines insect annotation by offering a graphical user interface for detecting and annotating the scanned collections. To this end, *Inselect* offers a object detection function based on conventional computer vision. However, manually correcting the automatically annotated objects is reportedly one of the bottlenecks in the scanning workflow [6]. Therefore, improving automatic object detection is an important step in accelerating this workflow and, hence, the digitization efforts. Also, *Inselect* does neither offer segmentation or classification [6], which might be useful in some application settings (e.g. compression and verification of labelling, respectively).
In many areas of computer vision, great progress has been made by substituting or assisting conventional processing methods by deep-learning (DL) based methods (e.g. [8, 9, 15]). Therefore, this work aims to study potentials ways of applying DL-based object detection and segmentation methods applied to the field of insect collection box digitization. Here, a special challenge arises from the very nature of the input images: Whereas popular datasets such as *ImageNet* [16] or *Common objects in context* (COCO) [11] are usually used at scales like 224×224 pixels or 800 pixels on the smaller side, respectively, here, input images are very large (e.g. more than 5,000 pixels for each size) to contain and reasonably resolute all of the many contained objects. Hence, the input size is very high but can not be significantly scaled down without losing the necessary resolution of the contained small objects. However, the input size is limited via hardware constraints (especially GPU memory). Hence, suitable model architectures need to be chosen and image resolution and, thereby, precision needs to be balanced against hardware constraints.

1.1 Datasets

The data available for this project were initially composed of 3 annotated collection boxes of 3 different butterfly species at high resolution. Further, multiple images of unannotated beetle collection boxes were provided. Ground truth masks were obtained as described in section 4.1.1, however, no category or species label could be obtained for most of the images. Therefore, proxy labels were assigned without any zoological verification of a correct splitting of the instances.
For all but 2 boxes the resolution of the individual instances was very low ($> 40 \times 15$ pixels per instance) and the images were of rather low quality. In some initial experiments this proved

species	count
Mesembryhmus purpuralis	251
Smerinthus ocellata	71
Acherontia atroposa	30
beetle proxy 1	51*
beetle proxy 2	39
beetle proxy 3	25
Trichotichnus	67

Table 1: **Counts of annotated instances per species.** For beetle proxy 1 (*) additional 21 instances in a high density area of the box were annotated for validation in the CPA experiments on coleoptera images (section 5.2).

to be insufficient to train a model for robust object detection and classification (e.g. results in figure 5(e)). Hence, only one crop from a low-resolution image ("Trichotichnus") and multiple crops from 2 higher-resolution images ($> 120 \times 40$ pixels per instance) was annotated. Overall object counts are given in table 1.1. To acquire additional training data, the publicly available data set of images from the *coleoptera* (beetle) family from [4] was used. Images were pre-cropped to individual instances facilitating the annotation, however, the images could not be used for validation or testing. To obtain some additional test images from *Inselect* [6] were used. For a beetles species, the results from *Inselect* were provided allowing for qualitatively bench-marking own results. For both, the *Inselect* and coleoptera dataset category but no segmentation labels were provided, so the latter were also obtained as part of this work (see section 4.1.2).

2 Related work

2.1 Model architectures

DL applied to segmentation tasks rapidly gained popularity in recent years. One of the novel architectures is the *U-Net* which won the ISBI 2015 challenge by a high margin [15] and became one of the standard architectures for semantic segmentation tasks. Due to the ease of implementation and the usually lightweight design of U-Net-based architectures, it is still widely used. Besides the pyramidal design of the encoding and decoding layer, it introduced the idea of long-range skip connections improving the internal feature representation and, thereby, the masking accuracy [15] (see figure 1).
However, U-Net is a plain (semantic) segmentation model and does not feature instance segmentation itself. Hence, it is only applicable for closed objects and cannot handle overlapping instances well. In the given task overlapping instances occur rather frequently, so real instance segmentation is preferable. Additionally, fully convolutional designs are desirable for the task as this allows to perform detection on different resolutions without needing to divide the images into single patches and recombine the prediction to obtain the final output. Further, given that insect size and zoom levels might vary, an architecture that can perform segmentation at different scales is desirable.
Therefore, *TensorMask* [1, 20] was chosen as the network architecture in this project. Treating the instance segmentation as a 4D tensor prediction in a dense sliding window approach, *Ten-*

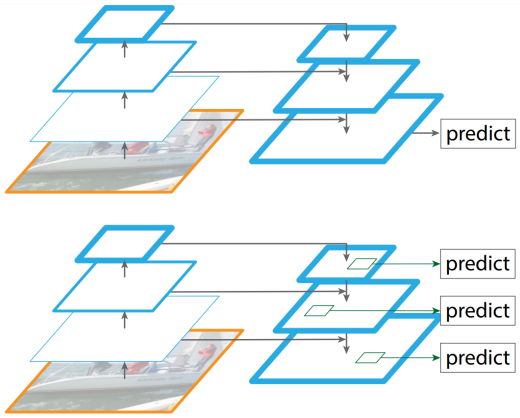


Figure 1: ***U-Net* (upper) and *feature pyramid network* (FPN) (lower) architectures.** Both models feature an encoding (left) and decoding (right) part which are connected via long-range skip connections. The green squares indicate independent predictions at each level of the FPN. Figure adapted from [10].

sorMask yields good results in dense object segmentation [1, 12]. As a standard state-of-the-art architecture for instance segmentation, *Mask-RCNN* [5] would have been another another possible choice for the task. *TensorMask* performances slightly worse on the COCO dataset than *Mask-RCNN* [1], however *TensorMask's* dense object segmentation approach was assumed to be able to handle high counts of small objects better than *Mask-RCNN*, which uses a separate *Region Proposal Network* for object detection in two-shot *detect-then-segment* approach [1, 5]. *TensorMask* was used in the default configuration with a *Feature Pyramid Network* (FPN) as backbone model. Similar to the *U-Net* architecture, FPNs allow for efficiently mapping abstract features in the decoder to the raw original objects in the encoder increasing the spatial accuracy of the predictions. Whereas *U-Net* performs mask prediction solely on the last layer of the decoder, the FPN additionally makes use of the intermediate layers of the decoder [1, 10, 15] (see figure 1). In *TensorMask*, the predictions of the different layers within the FPN are combined via convolutions in order to perform the final prediction. Thereby, larger objects are detected in deeper (i.e. specially smaller) layers of the FPN (in figure 1 on the top) whereas smaller objects are detected in shallower layers (in figure 1 on the bottom) [1]. Due to this design, it was assumed this architecture would be able to efficiently learn and segment especially challenging small-scale structures such as the insect extremities at different scales.

2.2 Copy-and-paste augmentation

In addition to the inherent challenges of the task of segmenting collection boxes posed by their appearances, in this study, both data availability and heterogeneity were rather low. In addition, only a fraction of the data was labeled and additional labeling was proofed to be very labor-intensive. Thus, sophisticated methods for leveraging the available data were required. Therefore, an approach called copy-and-paste augmentation (CPA) was introduced, whereby single instances are placed in front of a fixed background to generate an almost infinite amount of instance recombination into labeled images used during training. To this end, e.g. [2] models the background in order to place suitable objects and evaluates different pasting methods, whereas [3] shows that even plain random object pasting without the aim to archive

realistic images or inducing further variance by augmenting the individual instances drastically increases the performances in the COCO instance segmentation dataset. Also, [3] does not use augmentations of single instances before pasting. Of interest, improvements are particularly strong if only a small subset of the data is used [3] making this approach particularly interesting for the given task.
An implementation of CPA in python is given by [21]. However, this implementation only allowed for random object placements. Further, the implementation does not recalculate bounding boxes of partially obscured objects and, hence, might result in wrong ground truth data. Therefore, an own CPA implementation was made loosely based on [21] (see section 4.3).

2.3 Species prediction

Studies on classifying beetles (coleoptera family) have been conducted using cropped images and dedicated architectures in [4]. Due to a reported relatively low accuracy and high-performance variation between species in this work, the expected accuracy performing classification only as one of many tasks in the given setting was expected to be rather low as well. Hence, the focus was more laid on object detection.

3 Background

3.1 Model architecture

As no other backbone architecture was used, *TensorMask* implicitly refers to *TensorMask* with an FPN backbone and the tensor bipyramid as implemented in *detectron2* [20] in the following. Note, that in the FPN configuration *TensorMask* it is fully convolutional. Therefore, learned feature extraction can be translated between different input resolutions. This is important as input resolutions increased from the COCO pre-trained model [1], to task-specific fine-tuning, and, eventually, inference and testing in this work from 800, to more than 1200 and 2000 pixels on the smallest side, respectively. The model architecture, the loss functions, and, if not state otherwise, hyper-parameters were left unchanged compared to the *detectron2* implementation of *TensorMask* [1, 20], however the cutoff to ignore mask prediction was increased to 0.5. All models were trained from initial weights pre-trained on the COCO dataset obtained from the *detectron2 model zoo* (*6x* learning-rate scheduler) [1, 20].
The standard *TensorMask* loss function composed of individual losses for bounding-box regression, segmentation, and object classification was also left unchanged from the default implementation in all experiments. For visualization, exponential smoothing (7) with $\alpha = 0.75$ applied to the train losses.

$$l_i^* = \begin{cases} \alpha \cdot l_i + (1-\alpha) \cdot l_{i-1}^* & \text{if } i \geq 1 \\ l_0, & \text{otherwise} \end{cases} \tag{1}$$

3.2 Image calculations

For image intensity thresholding, Otsu [13] and "mimimum" adaptive thresholds [14] were obtained using the scikit-image implementation [19]. For mean average and Gaussian blur filters the implementations in scipy [18] and opencv [7] were used, respectively. For morphological dilation and erosion the opencv implementation [7] were used with a square kernel.

Alpha blending Let A and B be images and α be a mask image of equal size. Let

$$\forall\, i, j:\ 0\ \leq \alpha_{ij} \leq 1 \tag{2}$$

than the image $Blend_\alpha(A, B)$ is defined as

$$Blend_\alpha(A, B)_{ij}\ :=\ \alpha_{ij} \cdot A_{ij} + (1 - \alpha_{i,j}) \cdot B_{ij} \tag{3}$$

3.3 Performance metrics

COCO metrics Both, representation of annotations as well as performance metrics for instance segmentation tasks were based on COCO [11]. However, for evaluation, the AP_S, AP_M, and AP_L scores were not used as defined fixed object sizes in the scores neglect potentially differing input image resolutions, so if the image size is increased, a small object might fall in the medium or large category.

IOU and modIOU For mask annotation, the *Intersection of the Union* (IOU) of equally-sized images A and B were computed as

$$IOU(A, B) = \Sigma_{i,j} \frac{\mathbf{1}_{A_{ij}>0\,\wedge\, B_{ij}>0}}{\mathbf{1}_{A_{ij}>0\,\vee\, B_{ij}>0}} \tag{4}$$

whereby A_{ij} denotes the value of the ith pixel in j^{th} column.
Further, plain IOU was extended to a modified IOU (modIOU) ignoring discrepancies between images A and B at pixels on object edges. To this end, the condition for the intersection was weakened, so that a positive pixel A_{ij} in A in close proximity to a corresponding positive pixel $B_{\hat{i}\hat{j}}$ in image B is sufficient to form an intersection (i.e. $d((\hat{i}, \hat{j}), (i, j)) \leq \epsilon$, with d as euclidean distance and ϵ as heuristic spacial tolerance parameter). For implementation and also a more intuitive explanation, morphologically dilating mask B with kernel size ϵ' indicates presence of a positive pixel $B_{\hat{i}\hat{j}}$ in less than $\epsilon =: \frac{1}{\sqrt{2}}\epsilon'$ distance and, hence, the modIOU can be approximated as

$$modIOU_\epsilon(A, B) = \Sigma_{i,j} \frac{\mathbf{1}_{A_{ij}>0\,\wedge\, \exists \hat{i},\hat{j}:\, d((\hat{i},\hat{j}),(i,j))\leq\epsilon\,\wedge\, B_{\hat{i}\hat{j}}>0}\ +\ \mathbf{1}_{B_{ij}>0\,\wedge\, \exists \hat{i},\hat{j}:\, d((\hat{i},\hat{j}),(i,j))\leq\epsilon\,\wedge\, A_{\hat{i}\hat{j}}>0}}{\mathbf{1}_{A_{ij}>0\,\vee\, B_{ij}>0}} \tag{5}$$

$$\approx \Sigma_{i,j} \frac{\mathbf{1}_{A_{ij}>0\,\wedge\, dil_{\epsilon'}(B)_{ij}}\ +\ \mathbf{1}_{B_{ij}>0\,\wedge\, dil_{\epsilon'}(A)_{ij}}}{\mathbf{1}_{A_{ij}>0\,\vee\, B_{ij}>0}} \tag{6}$$

$$=: modIOU_{\epsilon'}(A, B) \tag{7}$$

whereby $dil_{\epsilon'}(A)$ denotes morphological dilation with kernel diameter ϵ'.

4 Solutions

4.1 Semi-automatic mask annotation

Due to complex shapes and high instance counts, manually labeling each instance was considered infeasible. Hence, semi-automatic approaches were used and are described in the following. Additionally, a custom script based on opencv [7] was written in order to convert the obtained binary masks to the COCO format [11] used in *detectron2*. Mask encoding was adapted to preserve topology (specifically background enclosed by beetle extremities) while maintaining compatibility to COCO (see details on GitHub).

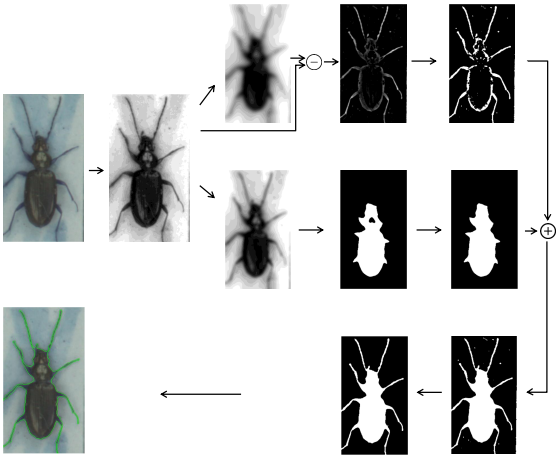


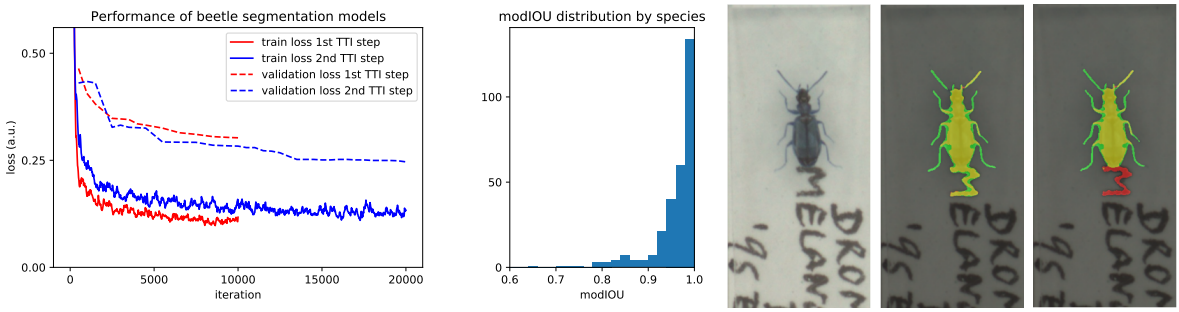
Figure 2: **Automatic beetle segmentation from crops.** The cropped raw image is first converted to grayscale and then blurred using a Gaussian blur (upper row). To segment the legs, the original grayscale image is subtracted from the blurred version and an Otsu adaptive threshold is applied. To segment the corpus, the grayscale image is blurred using a mean average filter and, consecutively, a "minimum" adaptive threshold is applied. Holes inside the corpus mask are closed and leg and corpus masks are combined using a logical OR (lower row). The final mask is obtained after applying morphological closing.

4.1.1 Annotation of ZFMK boxes

For labeling the fixed collection boxes, a semi-automatic approach was used leveraging the local contrast between the beetles and the background. To this end, images were converted into grayscale and pre-segmented using the "Auto Local Threshold" function of FIJI ImageJ [17]. Parameters were chosen individually for each processed image. The obtained masks were manually corrected e.g. to annotate missing extremities or separate fused masks of multiple instances.

4.1.2 Segmentation of cropped coleoptera images ([4])

Semi-automatic segmentation Due to the enormous size of the dataset, an approach requiring even mere manual correction of the obtained pre-annotated masks would not have been feasible. However, given that images are already cropped to single instances, the task of labeling can be reduced to sole segmentation. Further, the reduced image size resulted in more stable lightning and, hence, background-to-object contrast. Therefore, simple intensity-based thresholding proved to deliver reasonable results but frequently failed to segment thin legs or antennae. Making use of the distinctive shape of the beetles, extremities and corpora were segmented separately and the resulting masks are combined afterward as depicted in figure 2. Hereby, subtracting the original image from a blurred version enhances thin structures and edges to segment the legs, whereas a large median filter enhances the big and mostly convex corpus but eradicates the thin extremities. Further, closing the holes in the corpus mass is possible under the assumption of a closed body and morphological closing proved to improve segmentation quality by closing gaps between separated parts of the beetles such as between legs and the corpora.



(a) (b) (c) (d) (e)

Figure 3: **Beetle segmentation using *TensorMask*.** (a) Smoothed train (straight) and validation (dashed) loss of a TensorMask model in the first (blue) and second (orange) train-test iteration (TTI) step. (b) Class-wise average accordance (modIOU) of the mask predicted by the model obtained after the first TTI step and the automatic segmentation results. (c) Exemplary image in which the segmentation of a beetle close to text is improved after the second step of TTI compared to the model after the first step. (d) and (e) show the predicted masks of the model after the first and second step of TTI, respectively, in green, the results of the auto-segmentation in red, and the consensus area in yellow. Note, that in this case, both, auto-segmentation and the first TTI model failed to correctly segment the beetle from the text resulting in a higher modIOU score for the first TTI model than for the second TTI model, although the segmentation of the latter is obviously more correct.

This processing method yielded at least human performance in ca. 80% of the images and the errors were almost exclusively missing segmentation of extremities and, thus, easy and fast to spot. However, success rates varied widely between species and image lighting conditions.

Deep-Learning based segmentation To obtain more stable segmentation, a *TensorMask* model was trained to perform segmentation of the beetles. To this end, a set of 1,425 images for training (at least 2 of each species) and 291 for validation (one of each species) was obtained by either using a high-quality mask from the automatic segmentation pipeline or corrected versions. Obtaining this dataset took less than 4 hours of manual work.

The *TensorMask* model was trained for 10,000 iterations using default settings in *detectron2* [1, 20] (see figure 3a). To streamline model testing and find high-quality masks to grow the train set, IOU scores of the predicted masks and the automatically obtained masks (see above) were calculated. Given that a failure of both models at the exact same spot is unlikely, a high IOU score likely means correct results of both methods and, hence, implies that either one of the output masks is correct. However, visual inspection revealed a systematic discrepancy at the corpus boundaries resulting in a low score even when both masks were deemed correct by manual inspection. In turn, this artifact lowered the impact of highly relevant errors such as missing extremities. To compensate for this, the modIOU score was introduced and used for further evaluation.

To identify and consequently improve settings and instance types with low-performance species-wise modIOU scores were obtained (figure 3b). Visual inspection of examples from some of the species revealed three major sources of errors: (1) bugs overlapping with text and nails (figure 3c), (2) colorful (especially blueish) species, and (3) low-resolution species (presumably

condition	AP	AP_{50}	AP_{75}
1st TTI step	71.7	92.7	90.7
2nd TTI step	78.3	97.6	97.6

Table 2: Performance metrics of models obtained during train-test iteration for beetle masking on a test set composed of 291 species represented by one image each. Ground-truth masks were manually curated.

due to small biological size). Therefore, (1) text and nails were manually segmented from exemplary images and randomly placed inside train images. Also, (2) color shifts and (3) stronger reduction of image size were introduced as data augmentation. Further, the 50 worst performing species were over-sampled using masks pre-selected by high modIOU scores and manual selection or mask corrections. Following this, another *TensorMask* model was trained (see figure 3a) for 20,000 iterations and evaluated. Performance metrics are depicted in table 4.1.2.

Note, that in this setting, the unsupervised auto-segmentation is not a solid ground truth and, therefore, a low modIOU might be caused by an incorrect segmentation of the auto-segmentation rather a poor segmentation of the respective model. Thus, to compile the final dataset used for further experiments the following scheme was chosen: For each species, 30 examples were collected using manually annotated images from the second (1) and first (2) training and validation data sets, (3) images with a modIOU of $> 98\%$, and (4) the mask predicted from the model regardless of the modIOU score in the given order. Hence, the vast majority of the set is composed of images that were either manually curated or have high-level accordance between segmentation methods. For the remaining species with insufficient counts of high modIOU masks failure of the auto-segmentation was assumed.

4.2 Insect detection and segmentation

As a baseline approach, a *TensorMask* model was trained on the ZFMK box images. In order to improve generalization standard data augmentations including cropping and flipping were used.

4.3 Copy-and-paste augmentation

To increase the variability of the presented training samples beyond levels reachable via conventional full-image augmentations a CPA-based approach was implemented and evaluated (see figure 4). To make full use of the flexibility of the CPA each generated training image would optimally only be presented once.

CPA parameters As a baseline for the CPA a completely random distribution of objects within the images was evaluated (random CPA - R-CPA). Here randomly drawn instances are placed at coordinates sampled from a uniform distribution of valid x and y coordinates. However, objects are not placed if the overlap with existing objects is too high to avoid learning structures that would not be identifiable.

To mimic realistic collection boxes, a placement scheme in a grid-like layout was additionally implemented (collection-box imitating CPA - CB-CPA). Hereby, only objects from the same category (i.e. species) are positioned together. To control the object density, grid distances for

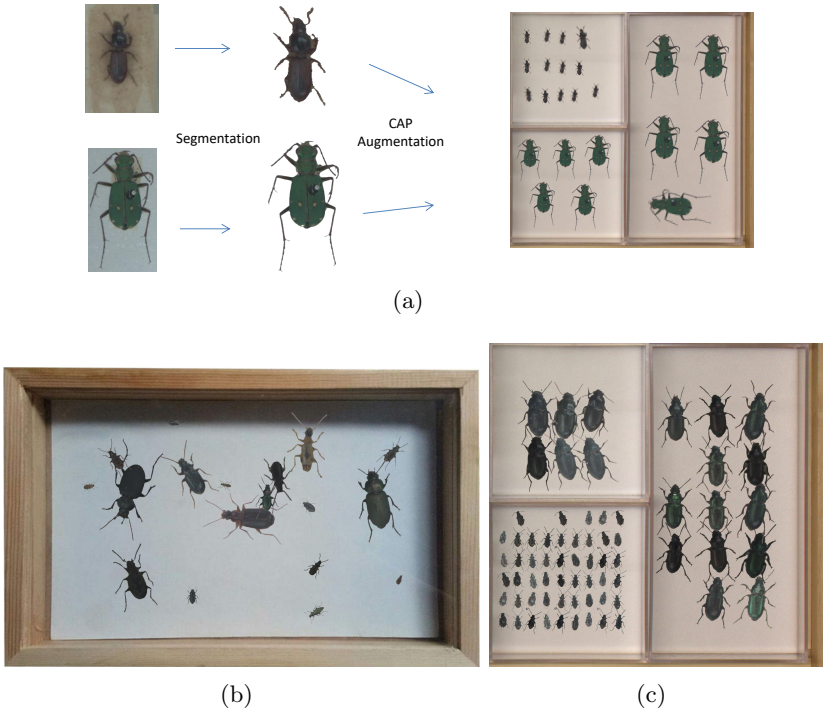


Figure 4: **Principle of copy-and-paste augmentation (CPA).** (a) For conducting a CPA a pool of instances annotated with mask is recombined to generate new images. Instances were recombined in randomly (R-CPA, (b)) and in a collection-box imitating pattern (CB-CPA, (c)).

each group were calculated via the mean images size and a randomly drawn factor. Also, random jitters are added to the exact grid position. To smooth edges between pasted objects and the background alpha-blending during pasting using a blurred version of the instance masks was also implemented and evaluated.
To increase the variability of the instances and potentially also achieve better generalizability to unknown species individual instances were augmented using random brightness-, contrast-, and Hue-shifts, optical distortions, and color PCA [16]. For R-CPA instances were additionally freely rotated, whereas rotations were limited to a range of $[-15°, 15°]$ to sustain a realistic appearance in CB-CPA. After CPA conventional data augmentations including color and contrast shifts as well as adding Gaussian Noise was added. In both CPA settings a variety of crops or masked regions from real collection boxes was used and instances were placed into rectangular spaces defined by the boxes to archive semantically correct images.

Caching strategy and instance scaling To avoid storing massive amounts of generated high-resolution training data an implementation generating training images in parallel to the actual model training was required. In turn, this imposed run-time constraints to less than the length of a single training pass. At the same time, loading, scaling, and, if specified, augmenting all instances before pasting into the target images was estimated to be too time-consuming even if distributed between up to four available cores. Hence, the instances are loaded, pre-processed, and cached in RAM before training although this reduces the data variability by re-using the applied instance augmentations. This could be partially overcome by additionally recalculating

figures/all_cat1.pdf

Figure 5: **Comparison of *TensorMask* models trained and tested on ZFMK collection boxes with and without using copy-and-paste augmentation (CPA).**

the pool of cached augmented objects during training, though this was out of the scope for this project.

Although the weights used for mask prediction are shared between the layers representing different scales in *TensorMask* [1], initial experiments showed that training poorly generalizes between layers (see figure 9). Hence, adjusting the scales of the presented objects during training to the rough size of the objects in the dataset is crucial. To avoid resizing slowing down the generation of training images, instances are cached on a pre-defined set of base scales. In the case of the CB-CPA, only instances at the same base-scale are placed alongside, whereas R-CPA ignores the base scale. To further increase variation in scales, instances were also rescaled as an augmentation.

Figure 5: (Image deleted due to copyright constraints): (a) and (b) performances metrics of models trained using CPA (red) and without using CPA (blue). (a) shows the smoothed training (solid line) and validation loss (dashed). The dotted line further shows the loss on the images the instances used for CPA were extracted from. (b) shows the AP of segmentation (solid) and object detection (i.e. bounding box prediction, dashed) on the validation set. (c) - (f) shows exemplary segmentation and object detection results from the models trained with (outer left) and without (outer right) CPA. Magnifications from instances masked during training and validation are shown in the middle. The magnified area is indicated as a black dashed box in the full-sized image. Note, that (d) is an annotated crop from (e) and yields completely different results on inference. Especially the model trained without CPA performs far worse and frequently miss-classifies the beetles at low scales in (e) as "Trichotichnus" from image (f).

5 Experimental evaluation

All evaluation was carried out in Colab Pro and Pro+ instances usually offering NVIDIA Tesla P100 with 16 GB of VRAM and up to 40 GB of RAM. Due to memory constraints, the images with at most 1600 pixels on the longer, 1300 pixels at the smaller, and not more than 150 instances. Mini-batch size was set to 1. In this setting, the model could be trained for 40,000 iterations in ca. 10-13 h.
For validation and testing, images were scaled to a maximum resolution of ≤ 3000 and ≤ 2500 pixels for the longest and shortest side, respectively.

5.1 TensorMask trained on ZFMK boxes

As a first attempt, the developed methods were evaluated on the ZFMK boxes. However, splitting the data into a train, test, and validation set proved to be particularly challenging: As images exclusively contained similar insects, splitting the images into the set would implicitly require the model to generalize to unknown species. Hence, the data was split between instances via masking out the respective instances in each set (see figure 10). As a baseline approach, a *TensorMask* model was directly trained on the obtained train images. Given the small size of the training set, standard data augmentations (random cropping at scales 0.2-0.9, flipping, color, and contrast shifts) were already applied to the baseline model. For the CPA, the pool sizes were heuristically determined and optimized, whereby training on a scale not covered by the presented instances drastically impaired training (see figure 9). Further, default instance augmentations and final augmentations were used.
For validation and testing, unseen instances inside the raw training images were used. In this experiment quantitative testing was only carried out on the annotated parts without simulating a full box. Tracking the validation loss during training revealed over-fitting of the non-CPA model to the train set (see figure 10(a)), which was expected given the high number of iteration in comparison to the low number of images and instances. For the CPA-trained model, the comparison of the raw training and validation loss is not complete given the different nature of both images. However, additionally tracking the loss calculated on the original images depicting the instances used for the CPA revealed the far better performance of the model after about 10,000 training iterations and, hence, also over-fitting to the instances extracted from the train images (see figure 10(a)). Overall, validation and loss and segmentation and bounding box metrics improved slower for the CPA training and did not reach the same level

train setting	segm. AP test	bbox AP test	segm. AP val.	bbox AP val.
fixed boxes	75.8	80	76.7	79.6
CPA	64.6	44.2	62.5	48.11

Table 3: **Performance comparison of models trained with copy-and-paste augmentation (CPA) and without using CPA (fixed boxes) on ZFMK boxes.** Validation (val) and test accuracy for segmentation (segm.) and object detection/bounding-box regression (bbox) after 20,000 iterations.

after 20,000 iterations, when training was stopped.
During testing, the CPA-trained model also obtained worse performance metrics than the baseline model (see table 5.1). However, the model trained without CPA failed to generalize to the zoomed-out view of the whole box. Whereas the CPA-trained model failed to learn the instance class "Trichotichnus" (see figure 5(f)), which was only contained in a very low-resolution image, the model trained without CPA performed very well in this class. This seemingly came at the price of miss-classifying apparently distinct species in other boxes as "Trichotichnus" if the instances were presented at a low scale (see figure 10(e)). Hence, the non-CPA model was able to learn this class exactly in given images without a robust transferability to other images or scales. Both models were also tested on images provided with *Inselect* (see figure 10), however, those results can not be quantified due to the lack of ground truth annotation. Optically, the predictions of both models appeared to be a lot worse on the *Inselect* images than on the ZFMK boxes.
Overall, these initial results showed that (1) training on images generated by CPA does generalize to the original collection box images, (2) CPA does over-fit to single instances after relatively long training requiring more training instances and augmentations for a longer training, and (3) CPA as implemented actually performs worse if training and testing are conducted based on the same raw images, but might generalize better towards unseen data especially if the scales differ from the ones present in training samples. To address instance over-fitting, the amount of data-augmentation, as well as the number of augmented copies, might be increased, however, a more thorough approach is simply obtaining more annotated data.
As the remaining ZFMK boxes (1) were low-resolution images causing some problems as previously described, (2) lacked species annotation and likely did not cover a large variety of instances classes (i.e. species), and (3) the annotation strategy for the collection box demands much manual annotation work, annotating more boxes was not feasible. Hence, further experimental validation of the CPA was conducted on the coleoptera dataset.

5.2 TensorMask training on coleoptera images

As the dataset consists of images cropped to individual instances, the latter needed to be recombined to generate images similar to the testing data. Hence, only the CPA approach was carried out, so the further experiments lack a non-CPA base-line comparison.
The training was conducted on the previously labeled instances from the coleoptera dataset and labeled parts of the beetle collection boxes were used for validation. To track the performance on different scales, (1) zoomed-out versions of the annotated parts of the collection boxes were recombined to archive a scale similar to the original full-sized images and were combined with (2) the raw zoomed-in annotated parts of the collection boxes. To also track performance on the two scales individually, performance metrics were additionally calculated

setting name	segm. AP crop	bbox AP crop	segm. AP whole	bbox AP whole
base	57.8	70.5	27.1	36.3
lower LR	60.5	69.4	28.2	34.6
R-CPA only	56.7	67.2	27.6	34.6
CB-CPA only	55.6	68.7	27.3	37.0
alpha blending	58.3	70.9	27.3	36.5
scale augmentation	58.5	70.2	27.2	36.0
small train set (1)	57.5	69.2	27.5	36.2
small train set (2)	58.4	69.9	27.4	36.2

Table 4: **Performance metrics of *TensorMask* models trained on the coleoptera dataset using copy-and-paste augmentation (CPA).** *TensorMask* models were trained using different CPA settings as depicted in table 6 for 40,000 iterations and tested on annotated crops (crop) and artificially created whole-box imitating images (whole). The COCO AP score was calculated for segmentation (segm.) and object detection/bounding-box regression (bbox).

on the two scales individually. As no significant difference in validation and test scores were observed in previous experiments (table 5.1), validation in test scores was assumed to be similar and generalizability was further controlled by using the unannotated images provided by *Inselect* as bench-marking.

In all following experiments, instances were scaled to a maximum species-wide mean size of 320, 160, and 80 pixels, whereby the maximum size of the entire image is between 1300 and 1600 pixels resulting in a relative base scale of ca. 0.25-0.2, 0.125-0.1, and 0.06-0.05, respectively. This was assumed to cover the scale of both types of test images at ca. 0.18 and 0.06 for zoomed-in and full-sized images, respectively. One (320 pixels), two (160 pixels), and three (80 pixels) augmented versions of each original instance were cached for the CB-CPA pool, whereas one augmented version of each instance was cached for the R-CPA pool if not indicated else wise.

As a base setting both, images from R- and CB-CPA were presented with equal probabilities resulting in solid detection accuracy. To investigate the effect of training with the collection-box imitations vs. with images generated completely randomly, *TensorMask* models were trained using solely CP-CPA and R-CPA, respectively. Whereas the R-CPA trained model archived a bit higher performance score on segmentation tasks, the CB-CPA performed better in object detection especially at low scale (see table 4 and figure 6), whereas the object detection accuracy was high in the early training phase (ca. 15,000 iterations) but sunk of afterward (see figure 6(d)).

Visualizing the prediction on the whole-scale collection boxes (see figure 7) shows a slightly higher number of detected instances (7(a) and (b)) but does not confirm any other apparent differences (c)-(j). Notably, the errors of both settings on large-scale images seem to be somewhat correlated and especially the instances presented in left and middle remain undetected by both, CB- and R-CPA trained models as well as by models trained on other CPA conditions (data not shown). Of note, the same instances were detected and segmented at a much higher accuracy if the image was zoomed in before inference (see figure 7 (c)-(f)), indicating that patch-wise processing would be yield better results than processing at full scale. The same effect was observed also in other images (see figure 7 (g)-(j)) and models (data not shown).

In an attempt to improve model performance, more realistic images were generated. To this

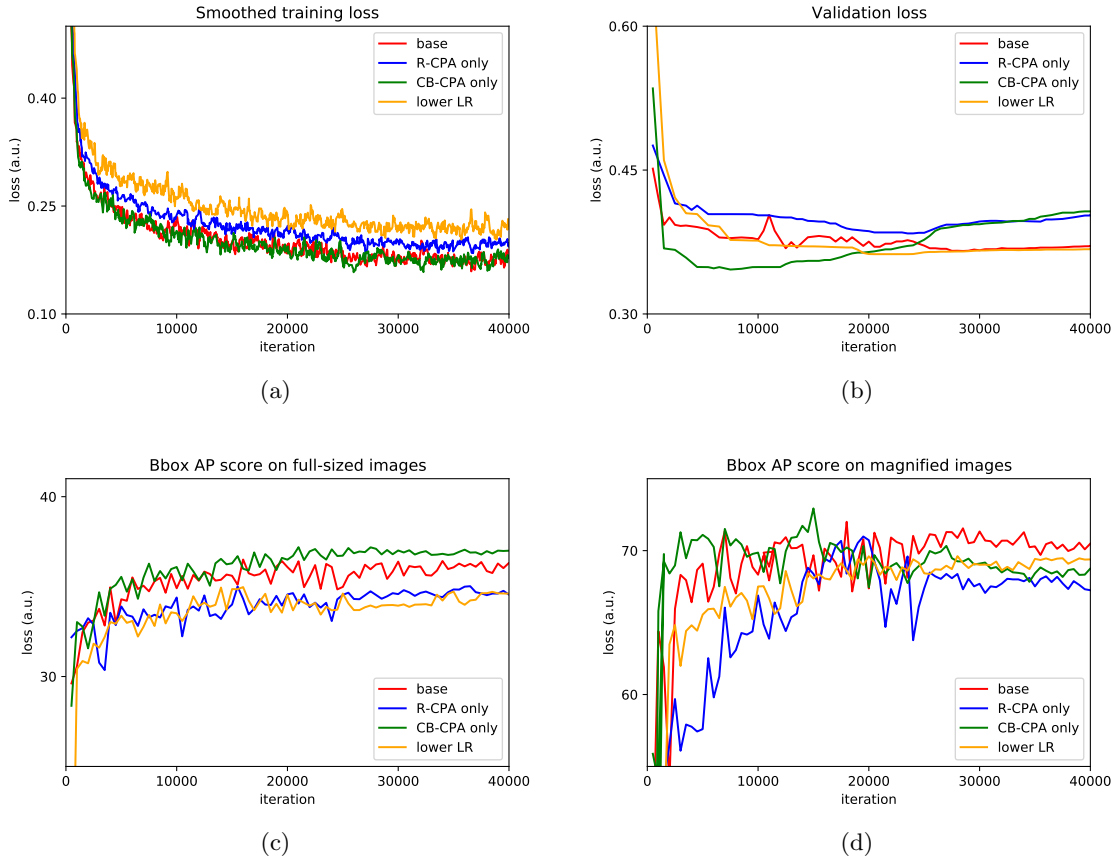


Figure 6: **Training progress of *TensorMask* model trained with different copy-and-paste augmentation (CPA) settings.** The models were trained using only collection-box imitating CPA (CB-CPA only), random CPA (R-CPA only), or a combination of both (base). Further, a model with decreased learning rate and increased batch-norm momentum (lower LR) was trained (see table 6). The training loss is calculated from images generated by the respective CPA setting, whereas the validation loss is calculated from fixed ZFMK boxes at magnified and full-image scales. (c) and (d) plots the COCO object detection/bounding-box regression (bbox) AP scores on only full-sized scale and magnified images, respectively.

end, the implementation of alpha-blending the pasted instances on the background images with a blurred version of the respective instance masks was tried. However, this increased performance only marginally and only on a larger scale. Further, scaling instances to scales in between the scales defined in the feature pyramid and adjusting the learning parameters was attempted. Whereas additional scaling did not affect the performance at a low scale, lowering the learning rate (LR) and increasing the batch-normalization (BN) momentum improved the segmentation accuracy while decreasing the object detection accuracy (see table 4 and figure 6).

In another experiment, training on a smaller training sample was simulated by using only 3 rather than 15 instances per species (small train set 1) and compensating for a small train set by increasing the number of augmentation per instance (small train set 2). Whereas performance

figures/output_box/big_stitch1.png

(a)

figures/output_rand/big_stitch1.png

(b)

figures/output_box/magn1.png

(c)

figures/output_box/big_stitch1.jpg

(d)

figures/output_rand/big_stitch1.jpg

(e)

figures/output_rand/magn1.png

(f)

figures/output_box/magn2.jpg

(g)

figures/output_box/big_stitch1.jpg

(h)

figures/output_rand/big_stitch1.jpg

(i)

figures/output_rand/magn2 (1)1.p

(j)

Figure 7: **(Image deleted due to copyright constraints): Exemplary prediction outputs for *TensorMask* model trained using collection-box imitating copy-and-paste augmentation (CB-CPA, left) and random copy-and-paste augmentation (R-CPA, right).** (a) and (b) show the outputs of the CB-CPA and R-CPA models, respectively on full-view collection boxes. (d) and (h) show a magnification of the lower half of the upper row in the second column and the upper left corner of (a), respectively, whereas (e) and (i) show the corresponding regions in (b). (c) and (g) show the prediction of the CB-CPA trained model on images cropped to the same sizes as in (d) and (h), respectively, whereas (f) and (j) show the corresponding predictions of the R-CPA trained model.

(a) (b) (c)

Figure 8: **Comparison of predictions from models trained with collection-box imitating copy-and-paste augmentation (CB-CPA, (a)) and random copy-and-paste augmentation (R-CPA,(b)) with object detection results from *Inselect* (c) [6].**

on large-scale instances dropped a bit for the first setting, this effect was compensated by the increased number of instance augmentations (see table 4). Hence, overall performance did not drop significantly implying that CPA, as implemented, is capable of learning from rather small datasets.
Overall, all models trained with the different CPA settings generalized to the *Inselect* data and outperformed *Inselect* on the depicted example image (see figure 8).

6 Discussion and outlook

In this work, I show that images of annotated single instances are sufficient to detect and segment beetles in collection boxes reliably at a large scale and at least moderately reliably on a small scale. Hereby, the failure of all the different trained models at certain instances remains elusive, however, given that almost the same errors occur at different CPA and training hyperparameter settings, it is likely caused by the composition of the training set not allowing to appropriately learn the affected instances as models trained using CPA from the fixed boxes were able to detect those exact instances. Of note, those instances do not show legs and other extremities. As extremities are usually present in the coleoptera dataset, the absence of those extremities might hamper the detection at this scale. Also, the exact taxonomic specification of the affected instances is unknown, so it might be a taxonomic group that was not present in the training set further decreasing the detection confidence. Hence, adjustments to the set of training instances might further increase the performance, and compiling an appropriate training data set considering specific features such as the presence of extremities as well as other potentially domain-specific knowledge might improve the composition of a more appropriate train set.
Still, I demonstrate that in general training mostly can be transferred to most unknown species and is mostly consistent between different origins of the image data, especially at smaller scales. Moreover, my results indicate that a moderate number of annotated objects (less than 900) are sufficient to achieve almost full accuracy even without sophisticated and potentially time-consuming instance augmentation. Also, my results indicate the potential for object

classification at least on a broader level, however, due to the lack of appropriate labeled data, this could not be finally proven.
As I demonstrate for beetles, images cropped to single instances can be annotated with minimal manual efforts and used for training a DL model capable of detection and segmentation in real drawers. Given the massive digitization efforts conducted in past based on tools like *Inselect* large-scale collection of single-instance images from different types of animals similar to the dataset used in this work are publicly available (e.g. from the National History Museum in London). Therefore, the proposed methods could be easily transferred to other insect or animal species and this work can be seen as a proof-of-concept to use CPA to leverage existing pre-annotated data to obtain more robust object detection in collection boxes.

References

[1] Chen, Xinlei ; Girshick, Ross B. ; He, Kaiming ; Dollár, Piotr: TensorMask: A Foundation for Dense Object Segmentation. In: *CoRR* abs/1903.12174 (2019). – URL `http://arxiv.org/abs/1903.12174`

[2] Dvornik, Nikita ; Mairal, Julien ; Schmid, Cordelia: Modeling Visual Context is Key to Augmenting Object Detection Datasets. In: *CoRR* abs/1807.07428 (2018). – URL `http://arxiv.org/abs/1807.07428`

[3] Ghiasi, Golnaz ; Cui, Yin ; Srinivas, Aravind ; Qian, Rui ; Lin, Tsung-Yi ; Cubuk, Ekin D. ; Le, Quoc V. ; Zoph, Barret: Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation. In: *CoRR* abs/2012.07177 (2020). – URL `https://arxiv.org/abs/2012.07177`

[4] Hansen, Oskar L. ; Svenning, Jens-Christian ; Olsen, Kent ; Dupont, Steen ; Garner, Beulah H. ; Iosifidis, Alexandros ; Price, Benjamin W. ; Høye, Toke T.: Species-level image classification with convolutional neural network enables insect identification from habitus images. In: *Ecology and evolution* 10 (2020), Nr. 2, S. 737–747

[5] He, Kaiming ; Gkioxari, Georgia ; Dollár, Piotr ; Girshick, Ross B.: Mask R-CNN. In: *CoRR* abs/1703.06870 (2017). – URL `http://arxiv.org/abs/1703.06870`

[6] Hudson, Lawrence ; Blagoderov, Vladimir ; Heaton, Alice ; Holtzhausen, Pieter ; Livermore, Laurence ; Price, Benjamin ; Walt, Stéfan van der ; Smith, Vincent: Inselect: Automating the Digitization of Natural History Collections. In: *PLOS ONE* 10 (2015), 11, S. e0143402

[7] Itseez: *Open Source Computer Vision Library.* `https://github.com/itseez/opencv`. 2015

[8] Krizhevsky, Alex ; Sutskever, Ilya ; Hinton, Geoffrey E.: Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems* 25 (2012), S. 1097–1105

[9] Lecun, Y. ; Bottou, L. ; Bengio, Y. ; Haffner, P.: Gradient-based learning applied to document recognition. In: *Proceedings of the IEEE* 86 (1998), Nr. 11, S. 2278–2324

[10] Lin, Tsung-Yi ; Dollár, Piotr ; Girshick, Ross B. ; He, Kaiming ; Hariharan, Bharath ; Belongie, Serge J.: Feature Pyramid Networks for Object Detection. In: *CoRR* abs/1612.03144 (2016). – URL `http://arxiv.org/abs/1612.03144`

[11] Lin, Tsung-Yi ; Maire, Michael ; Belongie, Serge J. ; Bourdev, Lubomir D. ; Girshick, Ross B. ; Hays, James ; Perona, Pietro ; Ramanan, Deva ; Dollár, Piotr ; Zitnick, C. L.: Microsoft COCO: Common Objects in Context. In: *CoRR* abs/1405.0312 (2014). – URL `http://arxiv.org/abs/1405.0312`

[12] Minaee, Shervin ; Boykov, Yuri ; Porikli, Fatih ; Plaza, Antonio ; Kehtarnavaz, Nasser ; Terzopoulos, Demetri: Image Segmentation Using Deep Learning: A Survey. In: *CoRR* abs/2001.05566 (2020). – URL `https://arxiv.org/abs/2001.05566`

[13] Otsu, Nobuyuki: A Threshold Selection Method from Gray-Level Histograms. In: *IEEE Transactions on Systems, Man, and Cybernetics* 9 (1979), Nr. 1, S. 62–66

[14] Prewitt, Judith M. ; Mendelsohn, Mortimer L.: The analysis of cell images. In: *Annals of the New York Academy of Sciences* 128 (1966), Nr. 3, S. 1035–1053

[15] Ronneberger, Olaf ; Fischer, Philipp ; Brox, Thomas: U-Net: Convolutional Networks for Biomedical Image Segmentation. In: *CoRR* abs/1505.04597 (2015). – URL `http://arxiv.org/abs/1505.04597`

[16] Russakovsky, Olga ; Deng, Jia ; Su, Hao ; Krause, Jonathan ; Satheesh, Sanjeev ; Ma, Sean ; Huang, Zhiheng ; Karpathy, Andrej ; Khosla, Aditya ; Bernstein, Michael S. ; Berg, Alexander C. ; Li, Fei-Fei: ImageNet Large Scale Visual Recognition Challenge. In: *CoRR* abs/1409.0575 (2014). – URL `http://arxiv.org/abs/1409.0575`

[17] Schindelin, Johannes ; Arganda-Carreras, Ignacio ; Frise, Erwin ; Kaynig, Verena ; Longair, Mark ; Pietzsch, Tobias ; Preibisch, Stephan ; Rueden, Curtis ; Saalfeld, Stephan ; Schmid, Benjamin u. a.: Fiji: an open-source platform for biological-image analysis. In: *Nature methods* 9 (2012), Nr. 7, S. 676–682

[18] Virtanen, Pauli ; Gommers, Ralf ; Oliphant, Travis E. ; Haberland, Matt ; Reddy, Tyler ; Cournapeau, David ; Burovski, Evgeni ; Peterson, Pearu ; Weckesser, Warren ; Bright, Jonathan ; van der Walt, Stéfan J. ; Brett, Matthew ; Wilson, Joshua ; Millman, K. J. ; Mayorov, Nikolay ; Nelson, Andrew R. J. ; Jones, Eric ; Kern, Robert ; Larson, Eric ; Carey, C J. ; Polat, İlhan ; Feng, Yu ; Moore, Eric W. ; VanderPlas, Jake ; Laxalde, Denis ; Perktold, Josef ; Cimrman, Robert ; Henriksen, Ian ; Quintero, E. A. ; Harris, Charles R. ; Archibald, Anne M. ; Ribeiro, Antônio H. ; Pedregosa, Fabian ; van Mulbregt, Paul ; SciPy 1.0 Contributors: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. In: *Nature Methods* 17 (2020), S. 261–272

[19] Walt, Stéfan van der ; Schönberger, Johannes L. ; Nunez-Iglesias, Juan ; Boulogne, François ; Warner, Joshua D. ; Yager, Neil ; Gouillart, Emmanuelle ; Yu, Tony ; contributors the scikit-image: scikit-image: image processing in Python. In: *PeerJ* 2 (2014), 6, S. e453. – URL `https://doi.org/10.7717/peerj.453`. – ISSN 2167-8359

[20] Wu, Yuxin ; Kirillov, Alexander ; Massa, Francisco ; Lo, Wan-Yen ; Girshick, Ross: *Detectron2*. https://github.com/facebookresearch/detectron2. 2019

[21] Yagfarov, Rauf: *RocketFlash CAP augmentation: First release*. 2021. – URL https://zenodo.org/record/4955371

Appendices

setting name	CB-CPA prob.	R-CPA prob.	LR	BN momentum	instances per species	additional modification
base	0.5	0.5	10^{-3}	0.9	15	-
lower LR	0.5	0.5	$2 \cdot 10^{-4}$	0.99	15	-
R-CPA only	0	1	10^{-3}	0.9	30	-
CB-CPA only	1	0	10^{-3}	0.9	30	-
alpha blending	0.5	0.5	10^{-3}	0.9	15	alpha blending added to CB-CPA
scale augmentation	0.5	0.5	10^{-3}	0.9	15	additional scaling of CPA instances
small train set (1)	0.5	0.5	10^{-3}	0.9	3	-
small train set (2)	0.5	0.5	10^{-3}	0.9	3	5-fold increased count of augmentations per instance

Table 5: **Names and parameters of different copy-and-paste augmentation (CPA) experiments.** In the scale augmentation setting, instances were additionally scaled by a factor uniformly drawn from $[0.85, 1.175]$ and $[0.6, 1.66]$ for CB-CPA and R-CPA, respectively. For the small train set (2), the number of augmented copies for each base image was increased by 5 fold compared to all other settings to reach the same overall pool size as the base condition. Abbreviations: CB-CPA - collection-box imitating CPA, R-CPA - random CPA, prob. - probability, LR - learning rate, BN - batch normalization.

figures/Appendix/scale_failure1.pdf

Figure 9: Detection inconsistency between scales. Predictions of a *TensorMask* trained on *(Image deleted due to copyright constraints): Acherontia Atroposa* instances at a relative scale of ca. 0.11 and 0.22 (largest instance size divided by largest image size) on an artificial test image composed of image crops showing instances at different relative scales. Instances in the test images are at relative scales of 0.082, 0.12, (upper row, left to right), 0.14, 0.16 (middle row, left to right), and 0.19 (lower row). The frame shows the size of the image as used for inference. In the the upper left image crop (i.e. at lowest scale) a label from a beetle species is assigned to the butterfly instance.

figures/Appendix/Bild11.jpg figures/Appendix/Bild21.jpg figures/Appendix/Bild31.png

(a) (b) (c)

(d) (e)

Figure 10: (Image deleted due to copyright constraints): Exemplary train (a), validation (b), and test (c) instances created from an annotated collection box crop. Predictions on benchmark image from *Inselect* of a model trained on fixed ZFMK boxes using copy-and-paste augmentation (CPA) (d) and without using CPA (e). For (e) the confidence threshold was lowered from 0.5 to 0.15 as no prediction reached a confidence of 0.5. All depicted predictions were classified as "Trichotichnus".