International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 12 (2020) pp. 239-247

Received: 5 April, 2020; Accepted: 23 May, 2020; Published: 20 June, 2020

Towards a Decision Support System for the Automatic Detection of Asian Hornets and Removal Planning

Diogo Braga[1,2] and Ana Madureira[1,2]

[1,2] Interdisciplinary Studies Research Center, Institute of Engineering - Polytechnic of Porto
R. Dr. António Bernardino de Almeida 431, 4200-072
Porto, Portugal
{1140499, amd}@isep.ipp.pt

***Abstract*: The rapid expansion of Asian hornets poses a high threat for the honey bee survival, as these invaders pray on them. Furthermore, they also pose a threat to people who are allergic, whose sting can lead to death. This study proposes a Decision Support System that uses Computer Vision techniques to automatically detect signs of Vespa velutina through images from GPS equipped camera. The goal of the system is to provide timely information about the presence of these invaders, allowing park managers and beekeepers to act quickly in removing the Vespidae. The proposed methodology obtained an 85% accuracy in the detection of V. velutina using the Mask RCNN architecture, enabling the system to perform detection at 3 FPS.**

***Keywords*:** Vespa velutina, Decision Support System, Computer Vision.

I. Introduction

On average, 13% of all floral visits belong to honey bees (i.e., western honey bees *Apis mellifera*), who are currently the world's most frequent pollinators of crops. There are around 200 economically important plants that require bee pollination for reproduction, and around 5% of plant species worldwide are exclusively visited by honey bees [1]. As stated by the UN Food and Agriculture Organization [2], more than 75% of the world's food crops rely to some extent on pollination for yield and quality. The European Commission highlights that pollinators provide vital ecosystem services to crops and wild plants, forming a key component of European biodiversity [3].

Yet, several threats to honey bees still exist worldwide [2], [4]–[7], of which the most recent one is the yellow-legged hornet (*Vespa velutina*), commonly referred to as the Asian Hornet. These were accidentally introduced into regions such as Europe from Asia [8], and their biological invasion raises significant problems for several sectors of Europe. *V. velutina* prey mainly on domestic and commercial honey bees, and just their presence is enough to increase mRNA expression of oxidative stress-related genes, catalase activity and lipid peroxidation, impacting honey bees negatively [9]. Not only this added stress can contribute to events such as Colony Collapse Disorder, but *V. velutina* can also be deadly to allergic people, and the invaders have rapidly risen an impressive size, especially in the west of Europe [8]. Even though signs of these invaders date to records as early as 2003, their expansion to countries such as Portugal, France and the UK was only registered starting from 2012 [10].

The western honey bees pollinate crop species that compose up to one-third of the average diet, worldwide [11]. Since the world population is expected to rise in the upcoming years, creating a demand for higher food production for a sustainable growth, the significant threat that the *V. velutina* pose to both bees and humans creates a demand for the development of solutions that can help experts to quickly and efficiently deal with these Vespidae. There is also special need for solutions that provide help regarding the destruction of early-stage nests [8], since these play a vital role in the produces provided to humans by ecosystem services.

Recent advances in Deep Learning (DL) and Computer Vision (CV) have shown potential applications in the automatic detection of bees and their respective health, such as detecting environment conditions and signs of *varroa destructor*, among others [11]–[16]. Furthermore, there have been recent developments in CV [17] which show interesting potential applicability for systems that can help with the detection of Asian Hornet presence.

The goal of this study is to provide and develop an architecture of a Decision Support System (DSS) that can help decision makers effectively deal with this Vespidae invasion. The architecture has the goal to detect the presence of *V. velutina* through images from GPS equipped cameras, where the DSS uses this data to provide insightful information about the time and location of the detected *V. velutina*, as well as suggestions for the effective removal of the invaders. Furthermore, the DSS must also be able to perform this detection as close to real-time as possible, with enough accuracy to reduce the efforts of alert revision and inspection by its users. The system is aimed to be used by park managers who want to ward off *V. velutina* and beekeepers who want to protect their hives from potential threats.

The remainder of this paper is structured as follows: section

II provides an overview of DL and CV, as well as a review of the most relevant and recent frameworks and work in the context of this study's scope; section III describes this study's methodology, including the data sets used, architectures and any design decisions towards the development of the proposed DSS; section IV details the results obtained in the trials conducted for the development of the object detection model, as well as a discussion and comparison of these results with the state-of-the-art; section V summarizes the major findings of this study and highlights future work to be conducted.

II. Literature Review

In this section, an overview of recent contributions in CV, ML and honey bee monitoring systems will be performed, with special focus to systems and solutions that aim to detect bees. At the time of writing, it was not possible to find any relevant literature regarding the automatic detection of *V. velutina*. As such, due to the resemblances between bees and wasps, the review mentioned previously will be used to assess the different approaches that can be used for the detection of asian hornets.

On the topic of bee health systems, recent contributions have been made involving ML algorithms and bee health monitoring. The work done by Kulyukin et al. [12] used 9110 audio samples, equally distributed by "Bee Sound", "Noise Sound" and "Cricket Sound", in order to monitor a bee hive. The approach used a Convolutional Neural Network (CNN) based architecture, was tested on the BUZZ1 and BUZZ2 [18] data sets and compared with other types of ML & DL algorithms. The authors concluded that DL can be used to monitor bees in a beehive and the CNN-based approach obtained an accuracy of 95.21% and 96.53% on the BUZZ1 and BUZZ2 data sets, respectively. The study in [19] used video and CNN for the automatic detection of honey bees in a hive. The data set consisted of approximately 11 hours of video footage, where a tag was placed on bees. The best F1-score value achieved in the study was of 0.686 and the authors concluded that the manual labeling provided by the tags may be insufficient for bee detection. Aiming at the detection of mites and *varroa destructor*, both [13] and [15] used CNN and ML algorithms to perform bee hive monitoring. The first study obtained an accuracy of 93% detection of *varroa destructor* using a training data set of 5000 artificially generated images, tested with different CNN configurations [13]. The second study used different light settings and a special camera setup, recording 1920x1080 images at 50-60FPS. The authors highlighted the challenge in detecting mites on moving bees or on bees where wings occlude the mites [15]. The work done in [20] provides a thorough explanation and exploration of the data set publicly available in [21]. In the approach, 2 CNN models were used to perform the classification of a bee's health and subspecies. The author of [20] provides a solid pipeline for modular training of the CNN models, including several methods to analyze and remove bias from the data set. Using the data set provided in [21], the author was able to obtain a best accuracy of 84.92% and 86.54% on the health and subspecies models, respectively.

In ML, DL concerns the development of computational models that are composed of multiple processing layers, which can learn representations of data with multiple levels of abstraction [22]. Each processing layer is composed by a set amount of non-linear modules that transform a specific representation at one level. Since DL can discover intricate structures in high-dimensional data, it can be applied in many domains of science, business and government [22].

Computer Vision can be defined as the field of science which relates to the automation of tasks that the human visual system can do [23]. CV comprises fields such as image segmentation, which concerns the partitioning of an image into a set of regions that cover it, obtaining meaningful information from the regions. The former differs slightly from object detection, as the latter's regions are bounding shapes (usually boxes), whereas image segmentation's regions can be composed from pixel-by-pixel representations [24]. From the literature review conducted for this study, the most promising architectures to be used in the development of the DSS are Mask-RCNN and Single Shot MultiBox Detector (SSD).

In the last decade, advances have been made to the CNN architecture, depending on the goal of its application (i.e., image classification, object detection, image segmentation). Some of the most significant contributions are as follows [17]: AlexNet, which won the 2012-ILSVRC competition (one of the most difficult challenges for image detection and classification, at the time); GoogleNet, which won the 2014-ILSVRC competition (introduced the concept of split, transform and merge blocks); ResNet, proposed by Microsoft, for the net training of 150 layers deep networks and DenseNet, which uses the idea of cross-channel connectivity. One of the most significant contributions is the Mask-RCNN architecture, which is a Recurrent Convolutional Neural Network that extends the Faster R-CNN – one of the best architectures for object detection and image segmentation [25]. Mask-RCNN implements a branch for predicting an object mask in parallel with the branch that performs bounding box recognition. With this architecture, Mask-RCNN can perform faster and simpler training, with only a 5 Frames Per Second (FPS) loss [25].

The SSD architecture was inspired by the anchors adopted in Multibox, RPN and multi-scale representation [26]. SSD functions similarly to You Only Look Once (YOLO). However, instead of fixed grids, SSD uses a set of default anchor boxes with different aspect ratios and scales. This approach is used to discretize the output space of the bounding boxes. By fusing predictions from multiple feature maps with different resolutions, the network can handle objects of various sizes [26], [27]. Comparing SSD to YOLO applied to images of 300x300, SSD outperforms YOLO with better accuracy at 59 FPS. With tuned settings, SSD can even outperform Faster R-CNN. However, SSD cannot handle small objects by default, requiring a better feature extractor backbone (such as a ResNet101) in order to achieve this goal [27].

The study in [27] has a particularly detailed and exhaustive comparison of several object detection frameworks such as Mask-RCNN, SSD, YOLO and Faster R-CNN, as well as insights on how these were implemented.

Table **1** summarizes the authors' findings when comparing the models on the VOC 2007, VOC 2012 and Microsoft COCO data sets.

Architecture	VOC 2007 Ranking	VOC 2012 Ranking	Microsoft COCO
Faster	3rd/4th/5th	7th	13th

R-CNN			
Mask-RCNN	8th	12th	3rd/4th
SSD	1st/2nd	2nd/3rd	2nd/10th/11th
YOLO	N/A	4th	12th

Table 1. Comparison of different object detection architectures. The table depicts the ranking of each architecture in the tested data set. Multiple rakings represent different configurations of the architecture [27]

As can be seen in Table **1**, it can be stated that SSD seems to be the overall best architecture when compared to the remaining architectures, having achieved top rankings. Likewise, it can be stated that although Mask-RCNN had poor performance in the VOC data sets, it seems to have a satisfactory performance in the Microsoft COCO data set.

III. Methodology

A. DSS Workflow and Dynamics

As stated previously, the goal of this work is to develop a DSS for the monitoring of Asian Hornet presence, which can provide insightful information to its users, such as park managers or beekeepers. To develop the DSS, the architecture defined in **Figure 1** is proposed. The main logic of the DSS consists in using a camera's image feed to detect signs of *V. velutina*. Since multiple cameras can be used, the system requires a scheduler that can determine which camera's image will be evaluated at a given time, like how an operative system schedules a task to a CPU. After the image is fed onto the classifier, an alert constructor (specific to each camera) will store the result of the analysis in a buffer and begin evaluation. If the alert constructor finds that the percentage of images stored in the buffer is greater than a given threshold, then an alert will be fired, and all images will be recorded in the database until the percentage of images of *V. velutina* decreases below the threshold value. Essentially, each alert constructor's detection buffer works like a sliding window, allowing the system to know when an alert must be fired, as well as when to start and stop recording detection footage. Finally, when the alert is fired, a notification is sent to the system's dashboard so that its users can review the detected footage and act accordingly.

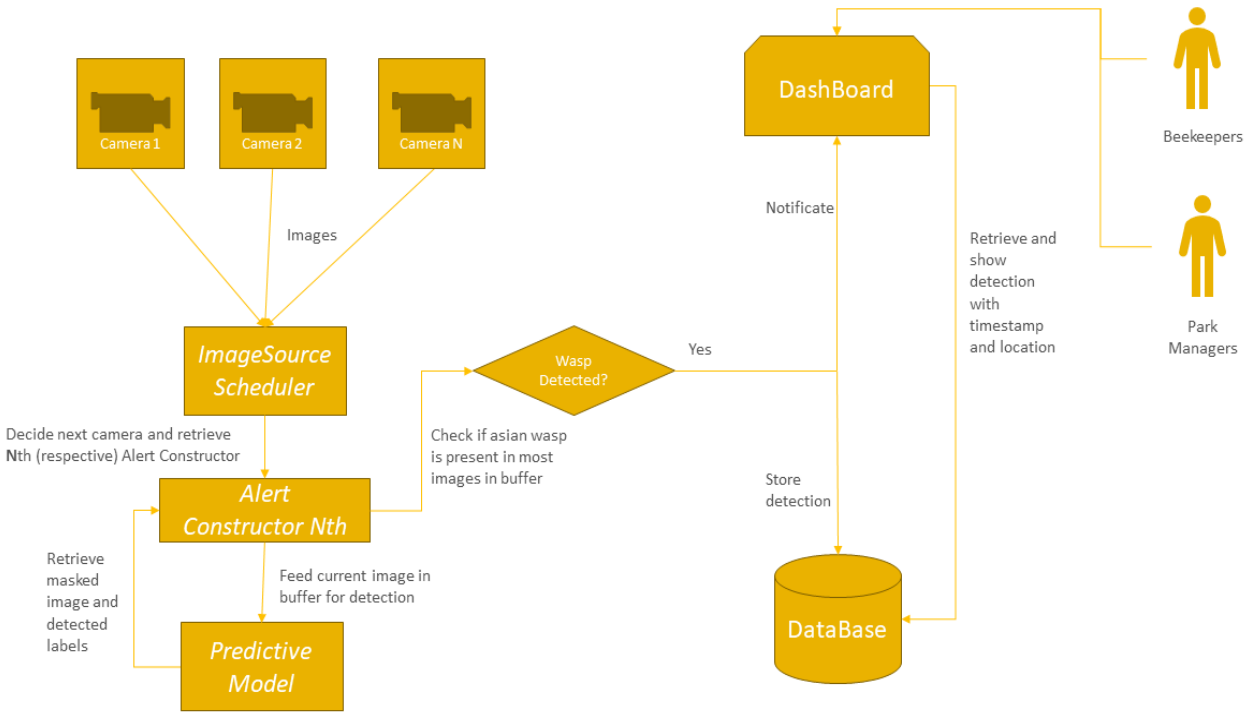


Figure 1. Diagram of the proposed DSS. Camera images will be fed onto a predictive model, following a scheduler's policy, to detect *V. velutina* and provide beekeepers and park managers with timely information.

On the topic of the system's scheduler, for this study, no needs for the type of job scheduler were found. As such, the implemented scheduler follows a First-In, First-Out (FIFO) approach, where each camera is inserted at the end of the queue after it's done processing, in a rotating fashion. Furthermore, since the classifier will always take a set amount of time generating the masked image, the detection buffer's length of each alert constructor is calculated by computing an estimate of the classification time. For the system to work properly, its administrator must specify the following constants:

- ***Sequence_seconds:*** the amount of time, in seconds, for which a camera must be detecting *V. velutina* signs until it is considered a valid alert. The previous is required because of the following constraint: if one were to be using a classifier with 90% accuracy, on footage recorded at 30 FPS, it would mean that roughly 3 out of 30 frames would have an incorrect prediction. As such,

to account for imprecision in the classifier, the alert is only fired if *V. velutina* signs are detected for a set amount of time.

- ***Global_alert_threshold:*** the percentage of images of which an alert constructor signals an alert. Similarly to the above, when accounting for the imprecision of a classifier, if in 10 frames only 1 does not have signs of *V. velutina*, then such case might have happened because of a classifier's inaccuracy (or even the wasp briefly being out-of-reach of the camera). However, as the alert should still be fired in such case, the possibility to configure the system's sensitivity to outlier data is also needed.

With the constants defined above, calculating an estimate for the detection buffer's length is possible using the formula described in (1).

$$buffer_{length} = ceil\left(\frac{sequence_{seconds}}{seconds_{per_{frame}}}\right) \quad (1)$$

Furthermore, when the system starts, it will run the detection mechanism for a set amount of time (default of 7 seconds) in order to get the average amount of time taken by the classifier for each detection (*seconds_per_frame*). As such, obtaining the desirable buffer length for a given system's configuration, in an automatic fashion, is possible.

The system was developed using Python, as the language possesses high versatility and compatibility with DL and software development frameworks, as opposed to R, which has some implementation incompatibilities due to third-party packages [28]. For the development of the DSS and respective dashboard, a web application approach was used, as it easily allows multiple users to view data without requiring any additional installation on computers.

Hardware	Specification
GPU	Nvidia Geforce GTX 1050 ti 4GB
CPU	Intel Core i7-8700 Hexa-Core 3.2GHz
Disk	540 MB/sec read speed 520 MB/sec write speed
RAM	32GB 2666 Mhz
Software	Python 3.6 Keras with Tensorflow GPU backend; tensorflow v1.13.1; keras v2.2.4; mask-rcnn v2.1; cuda v10.0.130; cudnn v7.6.0;

Table 2. Hardware specification of the environment used to develop the classifier and the DSS.

For this development, Django was chosen as the web development framework, since it is implemented in Python, reducing any overheads that could exist when calling models from frameworks of other languages such as Spring, .NET and Laravel. Table 2 provides the hardware specifications of the development environment of the DSS and the predictive model. Django can be considered as a high-level, free and open source Python web framework which encourages rapid development and clean, pragmatic design [29]. Django is used in production by companies such as Bitbucket, NASA, Udemy and Mozilla [30].

B. Accuracy Evaluation

For the purposes of evaluating the accuracy of a given model, it is important to discuss the topic of assessing a 'hit' or 'miss' of a given example. In object detection, the output of a given model is considered correct it the area of the predicted mask overlaps a desirable amount of the truth mask. The IoU provides the intersection of the masks (between 0 and 1), and the threshold commonly used is 0.5. However, the complex part of the evaluation algorithm resides in choosing the predicted mask that should be compared with the truth mask, as well as the expected behavior in the case of multiple overlapping predicted masks. For these reasons, a custom algorithm was implemented to evaluate the accuracy of the model. On the context of choosing the best candidate predicted mask, the custom algorithm chooses the predicted mask closest to the truth mask, using the Euclidean distance. The position used for the distance computation is obtained from the calculation of the center of the mask. Furthermore, on the case of the expected behavior in the case of multiple overlapping predicted masks, the following two alternatives were identified:

- **Consider multiple predicted masks as TP if their IoU is above the desirable threshold**, which translates to considering that the model predicted "correctly" that the detected object was in the given area, but did so multiple times (possibly meaning that the model fails to remember a given prediction);
- **Consider only one predicted mask as a TP, with the rest being FP**, which translates to considering that the model incorrectly predicted the extra masks and assumed that other objects/information close to the TP were the target object, when they shouldn't be (e.g. considering that a bee's leg is a bee).

Comparing both alternatives, the first alternative was concluded to provide a more optimistic assessment of the model's quality performance, albeit possibly less realistic. Likewise, the second alternative was considered to provide a less optimistic assessment, but more realistic. Taking into consideration the differences between the approaches and the potential impacts (both positive and negative) in obtained results, the second alternative was implemented in the custom algorithm, as to minimize benefits to the evaluated model's accuracy. *Table 3* shows the pseudo-algorithm, where, for each target class, the custom algorithm compares the predicted mask closest to the truth mask and, if the IoU is higher than the threshold, the mask is considered a true positive. This process is repeated until there are no remaining truth masks. Any remaining predicted masks are automatically labelled as a false positive.

```
1. def compute_confusion_matrix_image_data(self,truth_class_array,truth_mask_array,pred_class_array,pred_mask_array):
2.     ground_truths = self.pair_class_names_masks(truth_class_array,truth_mask_array)
3.     predictions = self.pair_class_names_masks(pred_class_array,pred_mask_array)
4.
```

```
5.  unique_classes = unique_list(truth_class_array)
6.  for clazz in unique_classes:
7.      has_gt_masks = False
8.      if clazz not in ground_truths:
9.          if clazz in predictions: #If mask does not exist in ground truths, but predictions has it, then it's a false positive
10.             self.matrix.add_value(clazz, 'fp', float(len(predictions[clazz])))
11.         else:
12.             #If not, then it is a true negative
13.             self.matrix.add_value(clazz,'tn',1.0)
14.     else:
15.         masks = ground_truths[clazz]
16.         if clazz not in predictions:
17.             #If predictions do not have any class, then it's a false negative
18.             self.matrix.add_value(clazz, 'fn', float(len(masks)))
19.         else:
20.             pred_masks = predictions[clazz]
21.             for i in range(0,len(masks)):
22.                 has_gt_masks = True
23.                 # if predictions has no targets for a class but there are still ground truths, then it's a FN
24.                 if len(pred_masks)==0:
25.                     #False Negative
26.                     self.matrix.add_value(clazz,'fn',float(len(masks)-i-1))
27.                 else:
28.                     index = self.closest_prediction_to_truth(masks[i],pred_masks)
29.                     pred_mask = pred_masks.pop(index)
30.                     iou = self.compute_intersection_over_union(masks[i],pred_mask)
31.                     if iou>self.iou_threshold:
32.                         #True Positive
33.                         self.matrix.add_value(clazz,'tp',1.0)
34.                     else:
35.                         #False Negative
36.                         self.matrix.add_value(clazz,'fn',1.0)
37.             #if predictions still has masks (extra), but ground-truths hasn't: FP
38.             if has_gt_masks and len(pred_masks)>0:
39.                 # False Positive
40.                 self.matrix.add_value(clazz,'fp',float(len(pred_masks))) #pred_masks should have the remaining masks
```

Table 3 - Pseudo-algorithm, in Python, for the assessment of an object detection model's accuracy.

C. Classifier Development

From literature review, it was concluded that both Mask-RCNN and SSD would be good candidates for the development of a classifier for this task. However, it was not possible to find a valid and working SSD architecture that would be robust enough to work on data sets that differ from its implementation. The only valid architecture found was the one contained in Tensorflow's Object Detection API [31], but the latest instructions for the correct implementation are dated and produce errors. Even after manual revision and correction, the training and evaluation trials of the SSD returned a 0% accuracy model. Due to time constraints and the complexity of the SSD architecture, it was not possible to perform a custom implementation of the architecture. Therefore, in this paper, the implementation, and trials of the SSD architecture will not be described.

For the development of the classifier, the data set was composed of images with and without bees and *V. velutina*. Since bees have a similar appearance to an Asian Hornet, a data set of these images was included, as to make sure the classifier would distinguish a *V. velutina* from a honey bee, therefore ensuring the usage of the DSS in environments with bee hives. The bee data set was extracted from Explore's honey bee live streams [32] and [33], which contain images from the perspective of the landing pad of two different apiaries and from inside a bee hive, respectively. For both live streams, images were extracted every 5 frames, totaling 2,750 images of honey bee footage for object detection. The wasp data set was obtained through manual search of *V. velutina* footage in documentaries and recording, such as in [34], as images of asian hornets aren't widely available, mainly due to the threat they pose (usually, there is an urge to remove them as fast as possible). In total, 130 hornet images were obtained. All images used were resized to 990x504 pixels. The images were manually annotated, in accordance to the following criteria: only bees and wasps that are at least 50% visible and whose quality is high (i.e., there is no blur associated to the bee) are annotated for object detection. **Figure 2** and **Figure 3** showcase an example of the annotations made following the previously mentioned criteria, using a custom-made tool for annotation.

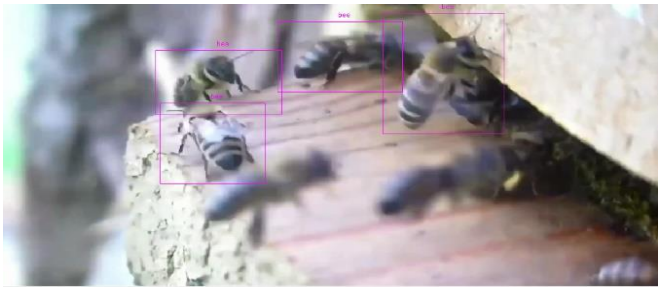


Figure 2. Example of annotations for object detection. In the image, bees whose quality is too low (e.g., blurry) are not annotated.

Figure 3. Example of an annotated V. velutina.

To maximize the performance of the algorithm, only 386 images were used for the object detection phase, which were manually annotated, instead of using an automated approach (i.e., Silver Standard). Of these, 49 correspond to nature-themed images with no bees. The empty images have the goal to further refine the model with examples of scenes where bees normally are present. Due to computational restrictions, all images and respective annotations were scaled down to 576x290 pixels. The evaluation metric used for the object detection task was the accuracy and average Intersection over Union (aIoU), which corresponds to the intersection of the predicted mask and ground-truth mask, over the union of the two masks. The accuracy was measured using IoU, with a threshold of 0.5 (i.e., an IoU higher than 0.5 is considered a true positive). The cross-validation technique chosen was the train/test split, as it still provides reliable results in low volume data, since, regarding object detection, cross-validation techniques provide minimal differences in accuracy [35]. Moreover, other cross-validation methods are impractical to use due to resource constraints. The train/test split was of 0.7/0.3. All splits were balanced in accordance to class distribution.

Regarding the Mask-RCNN, in order to perform the object detection of bees, the default configuration was used as a base and the changes stated in Table 3 were added to the training configuration.

Parameter	Value
NUM_CLASSES	3
IMAGE_MAX_DIM	576
IMAGE_MIN_DIM	320
IMAGES_PER_GPU	1
TRAIN_ROIS_PER_IMAGE	32
STEPS_PER_EPOCH	200

Table 4 - Mask-RCNN configuration for training the object detection model.

The TRAIN_ROIS_PER_IMAGE and IMAGES_PER_GPU were reduced due to computational constraints, as to lower the amount of VRAM required to train the model. *Table 4* shows the different values tested in trials for each configuration, with the goal of obtaining the best hyper-parameters for the object detection task. All trials conducted used the value of 500 as the random seed, for reproducibility purposes.

Parameter	Values
IMAGE_MAX_DIM	[576,460]
IMAGE_MIN_DIM	[320,256]
IMAGES_PER_GPU	[1,3]
TRAIN_ROIS_PER_IMAGE	[16,32,48,64]
BACKBONE	[‘resnet101’,’resnet50’]

Table 5 - Listing of all different values used for Mask-RCNN. Each trial tested a combination of the values specified in the table.

IV. Results & Discussion

Regarding the Mask-RCNN trials, each unique configuration was run for 40 epochs, since past this limit, the classifier did not seem to improve, as can be seen from **Figure 4**. From the trials conducted, the best results obtained for Mask-RCNN are the ones shown in Table 6, which corresponds to the configuration present in Table 5.

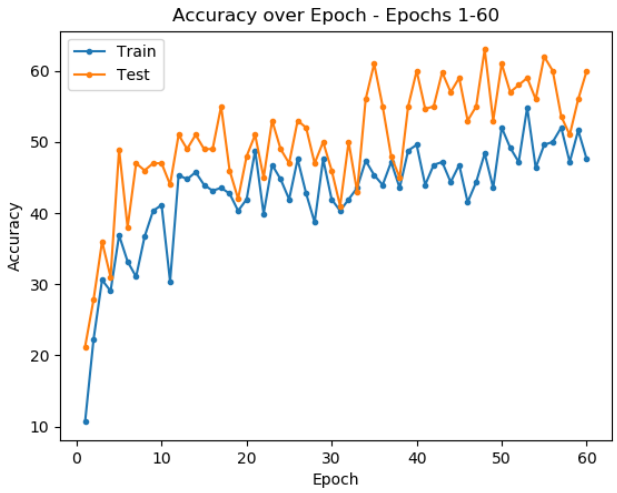


Figure 4. Evolution of train and testing accuracy over learning epochs. The accuracy change is approximately stagnant past epoch 40, as well as between epochs 20 and 40.

Parameter	Values
IMAGE_MAX_DIM	576
IMAGE_MIN_DIM	320
IMAGES_PER_GPU	1
TRAIN_ROIS_PER_IMAGE	64
BACKBONE	‘resnet101’
EPOCHS	40

Table 6 - Best configuration for the Mask-RCNN model.

Data set	aIoU	Accuracy
Train (Bee)	0.4444	47.44%
Test (Bee)	0.4512	53.75%
Train (Wasp)	0.5719	63.63%
Test (Wasp)	0.6827	85.00%
Train (All)	0.4614	49.60%
Test (All)	0.4975	60.00%

Table 7 - Best configuration's metric results. The metrics for all labels was obtained by combining the individual IoU of each sample and performing the test IoU>0.5

Using the best configuration obtained, the average time for the classification process, including the compositing of the resulting masked image, is approximately 0.36 seconds, which averages at 2-3 FPS. During trials, changes to the configuration were performed (e.g., changing resolution of the input image) to obtain a faster model, at the expense of the quality of the prediction. However, the increase in FPS was of 1-2 (totaling 4-5 FPS), with severe classification accuracy loss (of around 20%). As such, it can be stated that with Mask-RCNN, it is currently not possibly to obtain a faster classification process at the expense of accuracy, unless a more powerful hardware is used.

Comparing to related work, it can be stated that the approach defined in this study has interesting results. Regarding the bee detection, the developed classifier seems to have the same accuracy as other approaches. Regarding Asian Hornet detection, a direct comparison cannot be made, as there is no work found in the literature regarding the subject. However, when compared to bee detection accuracy, it can be stated that the classifier has a high accuracy in detecting signs of *V. velutina* through images, as evidenced by the 85% accuracy. The classifier exhibits satisfactory results, making its implementation in the DSS viable.

V. Conclusion

In sum, it can be stated that *V. velutina* can provide a great negative impact in both humans and bees, which in turn can cause a negative impact in the global economy and food production.

In this study, a methodology for the development of a DSS that can detect signs of *V. velutina* through cameras was proposed. The accuracy of the classifier in detecting asian hornets is satisfactory (85%), making the model and DSS viable for the automatic detection of signs of *V. velutina*. The model can still be improved since the detection accuracy for bees is still unsatisfactory (53%). Even though bee detection is not the scope of this study, the model could benefit from an improved detection process that can distinguish bees from *V. velutina*.

Future work consists in improving the classifier by providing more annotated data, as well as the further implementation of the SSD architecture, in order to ensure if the algorithm can generate a model that performs object detection with satisfactory results, at a higher frame rate.

References

[1] K.-L. J. Hung, J. M. Kingston, M. Albrecht, D. A. Holway, and J. R. Kohn, "The worldwide importance of honey bees as pollinators in natural habitats," *Proc R Soc B*, vol. 285, no. 1870, p. 20172140, Jan. 2018, doi: 10.1098/rspb.2017.2140.

[2] Food and Agriculture Organization of the United Nations, "FAO - News Article: Bees must be protected for the future of our food," May 20, 2018. http://www.fao.org/news/story/en/item/1132329/icode/ (accessed Nov. 21, 2018).

[3] European Comission, "Final Report Summary - STEP (Status and Trends of European Pollinators) | Report Summary | STEP | FP7," *CORDIS | European Commission*, Jan. 2015. https://cordis.europa.eu/result/rcn/176061_en.html (accessed Nov. 21, 2018).

[4] FarmMeetsTable, "Are Honey Bees Really Dying out? – Farm Meets Table," 2016. https://www.farmmeetstable.com:443/en/protecting-our-environment/2018/are-honey-bees-really-dying-out (accessed Nov. 21, 2018).

[5] A. Jacques *et al.*, "A pan-European epidemiological study reveals honey bee colony survival depends on beekeeper education and disease control," *PLOS ONE*, vol. 12, no. 3, p. e0172591, Mar. 2017, doi: 10.1371/journal.pone.0172591.

[6] S. Milius, "The mystery of vanishing honeybees is still not definitively solved," *Science News*, Feb. 13, 2018. https://www.sciencenews.org/article/mystery-vanishing-honeybees-still-not-definitively-solved (accessed Nov. 21, 2018).

[7] R. M. Underwood and D. vanEngelsdorp, "Colony Collapse Disorder: Have We Seen This Before?," p. 8, 2008.

[8] K. Monceau, D. Thiery, and O. Bonnard, "Vespa velutina: A new invasive predator of honeybees in Europe," Mar. 2014, Accessed: Nov. 12, 2019. [Online]. Available: https://www.researchgate.net/publication/258769112_Vespa_velutina_A_new_invasive_predator_of_honeybees_in_Europe.

[9] M. Leza, C. Herrera, A. Marques, P. Roca, J. Sastre-Serra, and D. G. Pons, "The impact of the invasive species Vespa velutina on honeybees: A new approach based on oxidative stress - ScienceDirect," vol. 689, Nov. 2019, Accessed: Nov. 12, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0048969719330839.

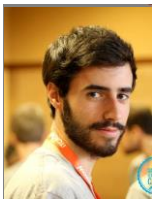
[10] Coloss, "Velutina – COLOSS," 2019. https://coloss.org/task-forces/velutina/ (accessed Nov. 12, 2019).

[11] A. Tiwari, "A Deep Learning Approach to Recognizing Bees in Video Analysis of Bee Traffic," *Grad. Theses Diss.*, Aug. 2018, [Online]. Available: https://digitalcommons.usu.edu/etd/7076.

[12] V. Kulyukin, S. Mukherjee, and P. Amlathe, "Toward Audio Beehive Monitoring: Deep Learning vs. Standard Machine Learning in Classifying Beehive Audio Samples," *ResearchGate*, Sep. 2018.

https://www.researchgate.net/publication/327478653_Toward_Audio_Beehive_Monitoring_Deep_Learning_vs_Standard_Machine_Learning_in_Classifying_Beehive_Audio_Samples (accessed Nov. 21, 2018).
[13] L. Chazette, M. Becker, and H. Szczerbicka, "Basic algorithms for bee hive monitoring and laser-based mite control," Dec. 2016, pp. 1–8, doi: 10.1109/SSCI.2016.7850001.
[14] I. Nolasco and E. Benetos, *To bee or not to bee: Investigating machine learning approaches for beehive sound recognition*. 2018.
[15] S. Schurischuster, S. Zambanini, and M. Kampel, "Sensor Study for Monitoring Varroa Mites on Honey Bees (Apis mellifera)," 2016.
[16] P. Amlathe, "Standard Machine Learning Techniques in Audio Beehive Monitoring: Classification of Audio Samples with Logistic Regression, K-Nearest Neighbor, Random Forest and Support Vector Machine," p. 57, May 2018.
[17] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A Survey of the Recent Architectures of Deep Convolutional Neural Networks," p. 62, 2019.
[18] Utah State University, "BeePi_Audio_Classification," 2019. https://usu.app.box.com/v/BeePiAudioData (accessed Aug. 15, 2019).
[19] M. Florea, "Automatic detection of honeybees in a hive," Sep. 2013. http://uu.diva-portal.org/smash/get/diva2:645634/FULLTEXT01.pdf (accessed Nov. 21, 2018).
[20] D. Pukhov, "Honey Bee health detection with CNN," Oct. 2018. https://kaggle.com/dmitrypukhov/honey-bee-health-detection-with-cnn (accessed Aug. 13, 2019).
[21] J. Yang, "The BeeImage Dataset: Annotated Honey Bee Images," 2018. https://kaggle.com/jenny18/honey-bee-annotated-images (accessed Aug. 13, 2019).
[22] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.
[23] RSIP Vision, "Difference Between Computer Vision and Image Processing," *RSIP Vision*, Jul. 27, 2015. https://www.rsipvision.com/defining-borders/ (accessed Oct. 09, 2019).
[24] L. Shapiro and G. Stockman, *CSE576: Computer Vision - Chapter 10*. 2000.
[25] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *ArXiv170306870 Cs*, Mar. 2017, Accessed: Aug. 15, 2019. [Online]. Available: http://arxiv.org/abs/1703.06870.
[26] W. Liu *et al.*, "SSD: Single Shot MultiBox Detector," *ArXiv151202325 Cs*, vol. 9905, pp. 21–37, 2016, doi: 10.1007/978-3-319-46448-0_2.
[27] Z.-Q. Zhao, P. Zheng, S. Xu, and X. Wu, "Object Detection with Deep Learning: A Review," *ArXiv180705511 Cs*, Apr. 2019, Accessed: Nov. 12, 2019. [Online]. Available: http://arxiv.org/abs/1807.05511.
[28] V. Kumar, "Python Vs R: What's Best for Machine Learning - Towards Data Science," Sep. 11, 2019. https://towardsdatascience.com/python-vs-r-whats-best-for-machine-learning-93432084b480 (accessed Jan. 01, 2020).
[29] Django Project, "The Web framework for perfectionists with deadlines | Django," 2020. https://www.djangoproject.com/ (accessed Jun. 04, 2020).
[30] Kalpit, "When To Use Django (And When Not To)," *Medium*, Jun. 12, 2019. https://medium.com/crowdbotics/when-to-use-django-and-when-not-to-9f62f55f693b (accessed Jun. 04, 2020).
[31] L. Vladimirov, "Training Custom Object Detector — TensorFlow Object Detection API tutorial documentation," 2018. https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/training.html (accessed Jan. 01, 2020).
[32] Explore, "Honey Bees Landing Zone Camera - live video of bees," May 2019. https://explore.org/livecams/honey-bees/honey-bee-landing-zone-cam (accessed Aug. 14, 2019).
[33] Explore, "Bee Cam - live camera inside of a honey bee hive," May 2019. https://explore.org/livecams/honey-bees/honey-bee-hive-cam (accessed Aug. 14, 2019).
[34] Animal and Plant Health Agency, "Close-up of Asian Hornet on the ground - YouTube," Mar. 24, 2015. https://www.youtube.com/watch?v=vxvYhgLNZsY (accessed Dec. 30, 2019).
[35] C. A. Ramezan, T. A. Warner, and A. E. Maxwell, "Evaluation of Sampling and Cross-Validation Tuning Strategies for Regional-Scale Machine Learning Classification," *Remote Sens.*, vol. 11, no. 2, p. 185, Jan. 2019, doi: 10.3390/rs11020185.

Author Biographies



Diogo Braga was born in Oporto, in 1996. He got is associate' degree in Informatics Engineering in 2017, and is expected to finish his Master's degree on July 2020, both at Institute of Engineering–Polytechnic of Porto (ISEP/IPP). He is a member of the Interdisciplinary Studies Research Center (ISRC) and has previously conducted work in the field of neurodegenerative diseases.

Ana Madureira was born in Mozambique, in 1969. She got his BSc degree in Computer Engineering in 1993 from ISEP, master's degree in Electrical and Computers Engineering–Industrial Informatics, in 1996, from FEUP, and the PhD degree in Production and Systems, in 2003, from University of Minho, Portugal. She became IEEE Senior Member in 2010. She was Chair of IEEE Portugal Section (2015-2017), and Chair/Vice-Chair of IEEE-CIS Portuguese chapter. She was Chair of University Department of IEEE R8 Educational Activities Sub-Committee (2017-2018). She is IEEE R8 Secretary (2019-2020). She is External Member Evaluation Committee of the Agency for Assessment and Accreditation of Higher Education - A3ES for the scientific area of Informatics of Polytechnic Higher Education (since 2012). Currently she is Coordinator Professor at the Institute of Engineering–Polytechnic of Porto (ISEP/IPP) and Director of the Interdisciplinary Studies Research Center (ISRC). In the last few years, she was author of more than 100 scientific papers in scientific conference proceedings, journals, and books.