DeepBees – Building and Scaling Convolutional Neuronal Nets For Fast and Large-scale Visual Monitoring of Bee Hives

Julian Marstaller[1]
julian.marstaller@online.de

Frederic Tausch[1,2]
fredetic.tausch@apic.ai

Simon Stock[1]
simon.stock@kit.edu

[1]Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
[2]apic.ai, Karlsruhe, Germany

Abstract

The decline of bee populations is a global trend and a severe threat to the ecosystem as well as to pollinator-dependent industries. Factor analysis and preventive measures are based on snapshot information. Information about the health state of a hive is infrequently acquired and remains labor-intensive and costly.
In this paper, we describe a system that enables near-time, scalable, and cost-efficient monitoring of beehives using computer vision and deep learning. The systems pipeline consists of four major components. First, hardware at the hive gate is capturing the in and out streams of bees. Secondly, an on-edge inference for bee localization and tracking of single entities. Thirdly, a cloud infrastructure for device and data management with near-time sampling from devices. Fourthly, a cloud-hosted deep convolutional neuronal net inferring entity-based health insights. This MultiNet architecture, which we named DeepBees, is the main focus of this paper. We describe the development of the architecture and the acquisition of training data. The overall system is currently deployed by apic.ai and monitors 49 beehives in Karlsruhe in the south of Germany.

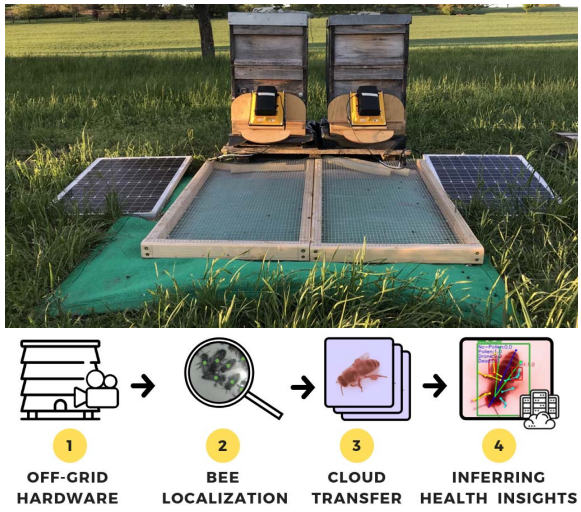


Figure 1: Hardware system and the proposed bee monitoring process. Bees are filmed at the hive entrance. Localization and tracking is performed on the edge. Sample streams of individual bees are transferred to the cloud and health data is inferred by DeepBees.

1. Introduction

In recent years, entomologists have been observing a global decline in the population of pollinating insects [20, 10]. This is of considerable concern since the global agricultural industry is heavily dependent on pollination [11]. One rather prominent representative of the species affected is the western honey bee (*Apis mellifera*). Unfortunately, the phenomenon cannot be traced to a single root cause. Recent research on honey bee population decline and colony collapse disorder (CCD) is suggesting a multitude of influencing factors [4, 12, 17, 18, 33]. Namely harmful pesticides, parasites, diseases, malnutrition and intruders. Furthermore urbanization [6] and intensive monoculture cultivation [14] are man-made causes. It is difficult to determine which of those factors ultimately lead to the death of a specific colony [36]. Deeper understanding of the ecosystem and human impact on nature is required as well as integrating this information in short and long-term decision-making. This leads to the need of a constant, low-cost surveillance of honey bee hives with the ability to automatically extract health-related insights. We developed a scalable, energy-efficient and non-invasive solution, using off-grid hardware with visual sensors as depicted in Figure 1. It continuously records the entrance of the beehive and evaluates the video material. Our system is detecting and tracking the insects on a locally running, Raspberry Pi-based, platform. Cropped image sequences of each insect are transferred to a cloud based computing system which

is not limited by computational and energy resources. The system architecture hence utilizes recent advances in edge computing [15, 28] to minimize workload and communication bandwidth costs. It also enables a flexible sampling of sequences based on current activity at the hive entrance, energy supply, and a general trade-off between costs and additional value due to enlarged samples. Also, information is processed at a large scale, from a variety of different locations within an ecosystem and made available near-time.

Our approach to offline bee tracking is based on the approach by Bozek et al. [2] who tracked honey bees on extracted honeycombs. This method is intrusive to the hives. Our proposed system is avoiding this issue by installing the system at the entrance of the beehive. Therefore, all bees entering and exiting can be monitored using a low-cost camera at 640×480 pixel resolution.

The proposed devices are currently deployed by apic.ai and collecting data from 49 beehives in and around the city of Karlsruhe in the south of Germany.

Having built this infrastructure, we now propose an architecture of a deep convolutional MultiNet as the primary analytical model to derive health-related insights.

2. Related Work

Pollinator decline is a global problem and intensively analyzed in the last decade. In Germany, Hallmann et al. [10] measured the biomass of flying insects through installed traps for 27 years. They registered a 76.7% drop of insect biomass, including wild bees from 1989 until 2016. Several approaches use less invasive sensor-based systems. Wario et al. used vision sensors to automatically detect the dance of bees and decode it [35]. The same authors noted that recognizing individuals visually alone is nearly impossible and equipped several bees with a marker around the head [37]. The system of Schneider et al. [27] uses RFID tags for monitoring. The drawback of such approaches is the modifications to the bees or their hive, which are required for those setups to work. Bozek et al. [2] tracked bees with CNNs on extracted honeycombs in large numbers and in close proximity to each other. They were able to detect bees through more than 720 video frames with over 375 thousand labeled bees. This technique enables easy and scalable monitoring on a individual level as well as bee counting at the entrance of the hive.

Few studies try to locate and detect pollen on images. In June 2017 Rodriguez et al. published a small dataset dedicated to classifying bees in pollen-bearing and non-pollen-bearing bees [23]. Later on, they used the dataset to test and evaluate several Convolutional Neural Network architectures [24]. In their approach images of bees were scaled to resolution of 180×300 pixels with RGB color channels. Rodriguez et al. [22] presented a particular interesting approach for detecting the pose of a bee using CNNs. They use GoogLeNet [31] to decode confidence maps for the left and right antenna as well as the tongue. Pereira et al. [19] developed a general framework for tracking the musculoskeletal system of insects. In particular tracking head, body, legs and wing positions of drosophila melanogaster.

We aim to solve multiple monitoring tasks in one network architecture. Therefore, we want to further investigate the feasibility of multi-task learning. Ge et al. [7] and Gebru et al. [8] showed that it is possible to transfer domain knowledge for solving a great variety of problems even if data for a certain challenge is sparse. The general idea is to use one common encoder with application specific decoders or classifiers. These approaches are currently adapted for several multi-task challenges like autonomous driving [32].

3. Concept

Solving the difficulty of multiple entities in an image and temporal consistency on the edge-device, simplifies the input of the cloud-based inference. The cloud-based CNN thus infers on single-entity images of bees. Since health indicators are versatile, we argue that a multi-task architecture is well suited to formulate learnable tasks and create annotated data for training.

3.1. DeepBees Architecture

The architecture shares a common encoder across all tasks as depicted in Figure 2. We name our proposal DeepBees, since it creates an extensive, feature-rich latent space representation (LSR) of bees based on a variety of tasks. Every trainable task adds additional features to the latent

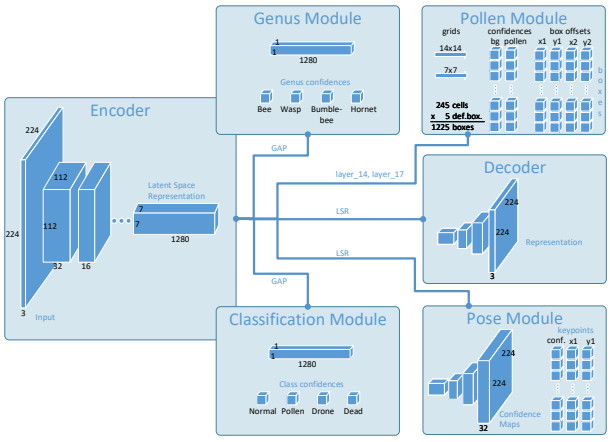


Figure 2: DeepBees, a MultiNet consisting of a shared feature extractor and modules for genus identification, pollen detection, unsupervised learning, pose estimation and classification.

space which can also improve the decision process of other tasks. We choose MobileNet-v2 with a width multiplier of

1.0 proposed by Sandler et al. [26] as the preferred encoder.[1] Next, we describe the output modules and tasks. They are restricted to tasks with available training data or feasible effort to collect and annotate such training data.

The **Genus Module** distinguishes between bees, wasps, bumblebees and hornets. This way, information about attacks, raiding or theft by intruders is collected. The relevant features for this classification are also useful to learn the general concept and shape of different types of insects. We use global average pooling (GAP) to reduce the latent space to a feature vector and the usual transformation with a dense layer and softmax to score probabilities for these four classes.

The **Pollen Module** detects pollen objects on bees. We use the Single-Shot-Multibox-Detector (SSD) approach proposed by Liu et al. [16] for this task. We score five default boxes for each cell on layer 14 and layer 17 of MobileNet-v2. By creating collections of detected pollen, we can create measures for the diversity of nutrition in the hive and aggregation over quantity measures. With this module, food shortages can be identified. We integrate a Tensorflow implementation by George Sung for this module [30]. To the best of our knowledge there exists no published work that attempts to detect pollen on bees using object detection.

The **Pose Module** module predicts the location of 32 relevant keypoints on the insect. Explicitly inferring the pose is beneficial in order to identify behaviour anomalies such as trembling after poisoning or infections. It can also be used to monitor grooming and hygienic behavior of the colony, since it is a defense mechanism against mites [1]. We closely follow the concept proposed by Pereira et al. for this task [19] by regressing confidence maps.

The **Classification module** scores probabilities for a total of four mutually exclusive classes. Worker bees with pollen, worker bees without pollen, drones and dead bees. While the pollen detection scores the box for single pollen, the classification infers on the entity-level. This is beneficial, since often only one pollen is visually present or is strongly blurred, if the leg is moved. The aggregation of the information sampled from the classification module can derive ratios about the population split, food availability w.r.t. colony size, as well as mortality ratios.

We also include a **Decoder** for structural learning. The architecture can hence also train the encoder on all unlabeled images.

We aim to add further tasks such as varroa detection, classification of specific diseases and activity scoring in the future as soon as training data is available.

[1] We evaluated VGG, Inception, ResNets, MobileNets and PNASNets at various sizes. MobileNet-v2 showed the best trade-off between performance and resource requirement. A detailed comparison goes beyond the scope of this paper.

3.2. Datasets

One benefit of a multi-task architecture is the flexibility to compile individual datasets for each task. It also allows for integration of existing datasets. Figure 3 displays samples from all datasets and classes. In Table 1, $|\mathbb{X}|$ summarizes the number of frames compiled in the dataset. $|\mathbb{X}_{train}|$ and $|\mathbb{X}_{test}|$ describe the training and testing sizes respectively.[2] We extended the VIA-Annotator by the Visual Ge-

[2] When further extended, we will include a validation set as well.

(a) Bees [13]

(b) Bumblebees [13]

(c) Wasps [13]

(d) Hornets [13]

(e) Drones

(f) Dead bees

(g) Bees with pollen

Figure 3: Exemplary images from all datasets.

ometry Group (VGG) [5] with a NodeJS-based back-end to interface with Google Storage for the annotation process. To reduce workload, we utilize sequence-wise aggregations of entities. Selecting images on sequence-level boosted dataset sizes by a factor of 20 (the average sequence length). We note that this practical methodology however introduces further bias in the data. Most importantly, it destroys the i.i.d. assumption of training data. E.g. successive frames of the same entity are highly stochastically dependent. This is why a random split between train and test data must indispensably assign sequence-wise and not frame-wise for correct performance measures. We propose to call this enlargement of training data **natural augmentations**.

Next, we elaborate on dataset-specific choices and insights for each task. For genus identification, the prototypes provided insufficient images of wasps, bumblebees and hornets. Manual search for these events was infeasible. Thus, we train this task solely on images from iNaturalist [13]. It is important not to mix it with bee images from the prototype, since this would lead to overfitting. Hence, the task is trained in another, closely related domain. Similar approaches such as multi-task domain adaptation by Gebru et al. have been proposed [9].

We integrated a small classification dataset by Rodriguez et al. and add box annotations. Furthermore, entity sequences with pollen were sampled from different devices and pollen boxes were drawn for each frame, creating a set of 5536 labeled images.

One issue with the classification module is class imbalance, especially with only a few samples of dead honeybees being dragged out of the hive. Every video or prototype contains specific calibration parameters such as background tint and slightly different focal lengths. We implemented two strategies to reduce the risk of decision boundaries based on non-task related information. First, for every sequence of an underrepresented class, we include a sequence for the other classes from the same video as well. Additionally, we introduce special augmentations to reduce potential bias. This includes the commonly used mathematical groups of rotation, shifting, shearing and flipping. To avoid tint bias, we utilize targeted color transformations based on Reinhard et al. [21]. This augmentation adjusts tint in order to match a target which we sampled from the overall training images. We used an OpenCV implementation for this augmentation [25].

Table 1: Overview of datasets sizes

Task	**Origin**	$\lvert\mathbb{X}_{train}\rvert$	$\lvert\mathbb{X}_{test}\rvert$	$\lvert\mathbb{X}\rvert$
Genus classification	[13]	8519	632	9151
Pollen detection	Ours	5238	298	5536
Pollen detection	[23]	629	85	714
Classification	Ours	9021	903	9924
Pose Estimation	Ours	191	38	229
Structural Learning	Ours	69387	15681	85068

The process of a systematic publication of the dataset as a potential benchmark is ongoing. Until this infrastructure is built and data quality is reviewed, please refer to data@apic.ai for early access.

3.3. Training

Training a multi-objective loss function imposes additional challenges. In particular the combination of pose estimation, object detection, two classification tasks and a decoder is to the best of our knowledge unprecedented. In this section we elaborate on procedures and best practices that we applied in order to stabilize this optimization problem.

First, we formally denote the total loss $\mathcal{L}_{\text{Multi}}$ as a summation of balanced task-wise losses as

$$\begin{aligned}\mathcal{L}_{\text{Multi}} =&\alpha_1\mathcal{L}_{\text{Cla}} + \alpha_2\mathcal{L}_{\text{Gen}} \\ &+\alpha_3(\mathcal{L}_{\text{Pol}}^{\text{Conf}} + \mathcal{L}_{\text{Pol}}^{\text{Loc}}) \\ &+\alpha_4\mathcal{L}_{\text{Pose}} + \alpha_5\mathcal{L}_{\text{Dec}} + \sum_{j=0}^{d}\beta_j\|\mathcal{W}_j\|_2^2\end{aligned} . \quad (1)$$

α_i describes the weight of loss $\mathcal{L}_i$. We also use a L2-regularization in all layers $j \in \{0, \dots d\}$ of the graph for all weights $\mathcal{W}_j$ and add it as $\sum_{j=0}^{d}\beta_j\|\mathcal{W}_j\|_2^2$. Classification tasks are trained using the standard cross-entropy loss, pose estimation and the decoder module use squared regression errors and the pollen detection loss consists of the box-classification and localization loss proposed by Liu et al. [16]. The prediction vector $\hat{\mathbf{y}}$ depends on the input image $\mathbf{x}$ and model parameters θ. The following tensor describes it for all five modules. p represent class probabilities, x_1, y_1, x_2, y_2 box coordinates and pk_x, pk_y the location of the maximum value (peak) in the confidence maps for each keypoint.

$$\hat{\mathbf{y}}(\mathbf{x},\theta) = \begin{bmatrix} \frac{(p_{\neg\text{Pollen}}, p_{\text{Pollen}}, p_{\text{Drone}}, p_{\text{Dead}})}{(p_{\text{Bee}}, p_{\text{Wasp}}, p_{\text{Bumblebee}}, p_{\text{Hornet}})} \\ \hline \begin{bmatrix}(p_{\text{Pollen-Box}}, x_1, y_1, x_2, y_2)^{(1)} \\ (p_{\text{Pollen-Box}}, x_1, y_1, x_2, y_2)^{(2)}\end{bmatrix} \\ \hline \text{confmaps}^{(1,\dots,32)}, \\ \begin{bmatrix}(pk_x, pk_y, pk_{\text{conf}})^{(1)} \\ \vdots \\ (pk_x, pk_y, pk_{\text{conf}})^{(32)}\end{bmatrix} \\ \hline \hat{\mathbf{x}} \end{bmatrix} . \quad (2)$$

We used task-wise performance metrics to evaluate each task, optimize hyperparameters and calculate a performance metric for the multi-task net. It weights each task equal as

$$\mathcal{P}_{\text{Multi}} = \frac{1}{4}\big(\text{AUC}_{\text{Cla}}+\text{AUC}_{\text{Gen}}+\text{AP}_{\text{Pol}}+(1-\text{AD}_{\text{Pose}})\big). \quad (3)$$

Classification tasks use the Area Under Curve (AUC) of the receiver operating characteristic as their evaluation metric. Pollen detection uses the average precision of box estimates at a 0.5 intersection over union (IOU). The pose estimation is evaluated by the average distance (AD) between predicted and actual keypoints. For a total of K keypoints, this is

$$\text{AD}_{\text{Pose}} = \frac{1}{K}\sum_{j=1}^{K}\frac{1}{|\mathbb{X}_{\text{test}}|}\sum_{\mathbf{x}\in\mathbb{X}_{\text{test}}}\frac{1}{d}\,\|\widehat{pk}^{(j)} - pk^{(j)}\|_2 \quad (4)$$

We scale the difference between actual and predicted keypoints by d, which represents the diagonal of the image and thus scales the error to an interval between zero and one. Additionally, we tracked Multiscale Structural Similarity (MS-SSIM) for the image reconstructions of the decoder during training based on Wang et al. [34].

We chose **joint training** as the training methodology instead of alternate training, because we explicitly consider all tasks in every parameter update. Therefore, we feed a heterogeneous mini-batch to the model. We determine the proportion of observations from task datasets with random uniform sampling. Since an observation only contains annotations for one task, the loss and gradients of the other tasks cannot be calculated for that particular sample. In practice we therefore multiply the task-losses with an indicator ("mask"), that is 0 if there exists no annotation for the task and 1 otherwise.[3]

We test several **parameter update strategies**. First, we update model parameters θ using the regular stochastic **gradient descent** with gradients calculated from the combined loss $\mathcal{L}_{\text{Multi}}$ and learning rate η:

$$\theta \quad \leftarrow \quad \theta - \eta\nabla_{\mathcal{W}}\mathcal{L}_{\text{Multi}}. \quad (5)$$

We test two approaches that use gradient normalization. In the **global norm** scenario, we clip and normalize the whole gradient vector over the total loss:

$$\theta \quad \leftarrow \quad \theta - \eta\frac{1}{\|\nabla_{\mathcal{W}}\mathcal{L}_{\text{Multi}}\|_2}\nabla_{\mathcal{W}}\mathcal{L}_{\text{Multi}}. \quad (6)$$

This way, we try to keep parameter updates within a fixed range. In the last approach we form gradients on each task first, perform the normalization on each of these vectors and then combine them as an average. We denote this approach task-wise **Grad Norm** as proposed by Chen et al. [3]:

$$\theta \quad \leftarrow \quad \theta - \eta\frac{1}{\mathcal{T}}\sum_{t=1}^{\mathcal{T}}\frac{1}{\|\nabla_{\mathcal{W}}\mathcal{L}_t\|_2}\nabla_{\mathcal{W}}\mathcal{L}_{\text{t}}. \quad (7)$$

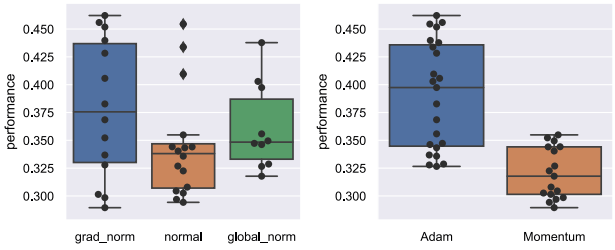
This way, the signal to the parameter update from each task should be represented equally. We can also combine the gradient normalization approaches with balance updates.

[3]Masked gradients also need to be excluded from moment estimates of optimizers

4. Experiments

4.1. Hyperparameters

To determine the right training setup, we first perform a random search with a total of 40 models trained for 3000 steps each and a batch size of 32. We see the results in Figure 4. The left plot depicts the gradient normalization



(a) Gradient Normalization. (b) Optimizer.

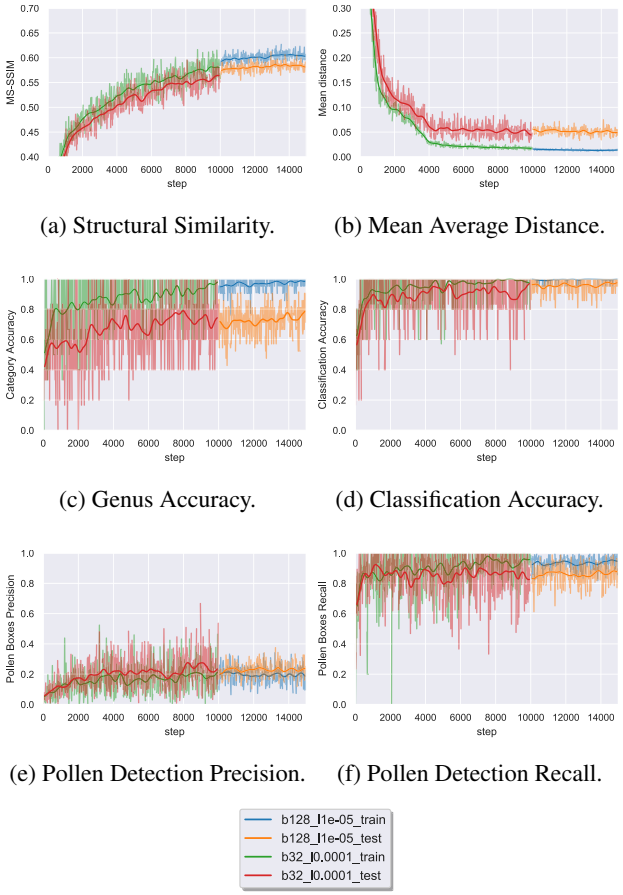
Figure 4: Random search on training hyperparameters.

strategies. In fact, using the global or the task-wise gradient norm (grad_norm), establishes better conditions for learning compared to the normal gradient descent setting. The Task-wise gradient norm achieves the highest performance and median and is used for the final training. In the second plot, we compare two different optimizers Adam and Momentum. Since Adam strongly outperforms Momentum, keeping individual learning rates is very useful in a joint training setting.

To mitigate signal differences from the hypersurface of each individual loss, we also trained each task individually for 3000 gradient updates and normalized the task loss for the multi-task training by the final training loss from the single-task setting. We keep this normalization fixed because the dynamic balancing of task losses did not work well with Adam.

4.2. DeepBees Training

We trained the final MultiNet in two stages. In the first stage the batch size is fixed to 32 and the learning rate to 0.0001. Therefore, the effective batch size for each task at that stage is $\frac{32}{5} \approx 6.4$ on average. The performance of pollen detection remained low with these batch sizes due to high noise in the box coordinate regression. In the second phase we decrease the learning rate to $1e^{-}5$ and increase batch size to 128, yielding an effective batch size of 25.6 on average for each task. Increasing batch size at a later stage is a finding by Smith et al. [29] and improved pollen detection a lot. Decreasing the learning rates at the end of optimization is also a common technique seen in literature. Figure 5 depicts various metrics and their changes on the training and test set. In the first chart, we see the MS-SSIM of original and reconstructed images. For both phases the similarity



(a) Structural Similarity. (b) Mean Average Distance.

(c) Genus Accuracy. (d) Classification Accuracy.

(e) Pollen Detection Precision. (f) Pollen Detection Recall.

Figure 5: Metric changes on train and test set during the optimization of the MultiNet.

score increases. The second Figure 5b tracks the average distance error of the pose estimation module. Due to the small number of labeled pose images, they are more frequently sampled compared to other images from other tasks and convergence is faster. The prediction accuracy of the genus module is shown in Figure 5c. It measures the percentage in which the predicted class matches the actual class in relation to all predictions. We notice a larger gap between test and training accuracy. Given the low signal-to-noise ratio of the dataset, this is coherent. In the second phase the training accuracy reaches its limits at close to 100%, while the test accuracy converges at approximately 70%. We can compare this measure to the accuracy of bee classification in Figure 5d. The classification task is learned faster and achieves accuracy above 90%, even for the test set. We also expect the AUC to reflect on the performance difference between the tasks. Figure 5e and 5f show precision and recall of bounding boxes of the pollen detection before post-processing with Non-Maximum Suppression (NMS). The task generally has good recall but struggles with the precision. Due to motion blur and occlusion, the presence

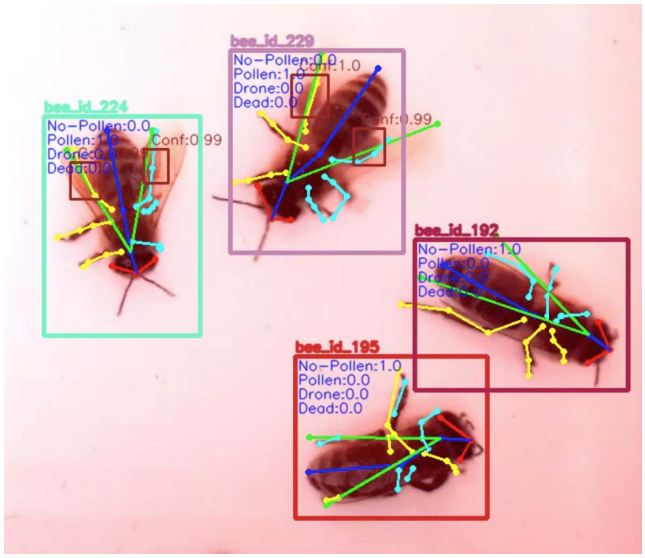


Figure 6: Exemplary outputs of DeepBees.

of pollen is often difficult to assess. The localization, on the other hand, is less challenging since pose features are explicitly learned.

A sample model output on test videos is shown in Figure 6. The net analyzes each detected bee individually.

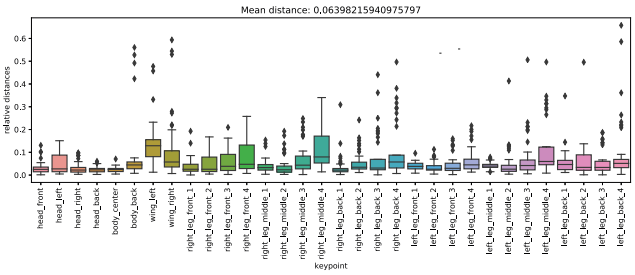
4.3. Evaluation

In Figure 7 we have collected the performance evaluations of every module. The variance in the average distance between estimated and actual keypoint is higher for the wings and outer points of the limbs since they are more difficult to score. The genus module struggles with hornets (yellow line), but manages to identify the other classes. In comparison, the classification module works well on all classes. The AP for the object detection task visualizes the trade-off between precision and recall of pollen detection. In Table 2 we compare the final performances of DeepBees with best performances acquired in a single task setting. Single tasks are trained for 6000 steps on 32 batches using Adam such that they saw the same amount of training data as DeepBees. Especially the classification task profits from multi-task learning. Genus and pose show similar performance levels while pollen detection worsens. We interpret that the features of the motion apparatus for pose estimation are on the one hand useful for pollen detection, but also increase additional noise when deciding if pollen is present or not. Overall the performance is better and while sharing the same sized latent space representation for all five tasks instead of customizing it for only one task.

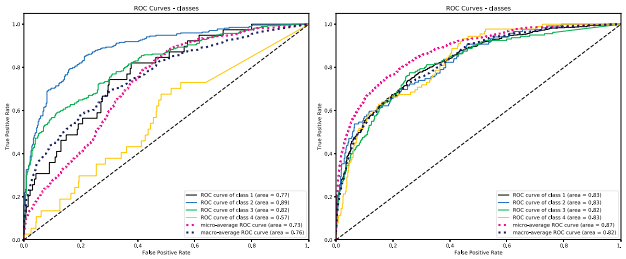
Major benefits also come in terms of deployment costs. Combining four insightful inferences to one helps to reduce power consumption and maintenance of the model with no performance disadvantage. We see this as the strongest

Table 2: Task performance comparison for individual training (Single-Task) and DeepBees.

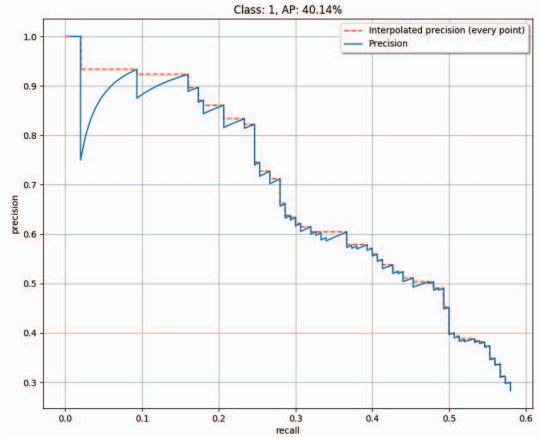
Model	$\mathbf{AUC_{cla}}$	$\mathbf{AUC_{gen}}$	$\mathbf{AP_{pol}}$	$1-\mathbf{AD_{Pose}}$	$\mathcal{P}_{\mathbf{Multi}}$
Single-Task	70.34%	76.05%	**46.64%**	90.12%	70.78%
DeepBees	**82.41%**	**76.19%**	40.14%	**93.61%**	**73.08%**



(a) AD of Pose Module.



(b) AUC Genus Module. (c) AUC of Classification.



(d) AP of Pollen Detection.

Figure 7: Performance Metrics of DeepBee modules.

achievement of DeepBees. The model is now deployed as a Tensorflow serving and analyzes data from the 49 honey bee hives.

5. Conclusion

In this paper, we shared an approach to efficiently model a system as well as a deep learning architecture to automate and scale honey bee monitoring.

We emphasized the practical added value of a multi-task learning approach being flexible due to its modularity and allowing for a cost-benefit adjustment in sampling and analyzing sequences from hives. We collected and integrated data such that all tasks could be trained and deployed at a suitable quality.

We also showed that when trained properly, the multi-task model does not worsen performance. We hope to see the latent space of DeepBees further improved in the near future with more health-related extensions. Especially direct mappings to activities, diseases or mite infestation remains a challenge due to lack of data. Defining temporal architectures could also further improve results. We are optimistic that a large scale collection of data from a variety of hives will capture the relevant and necessary amounts of data to learn and automate health-related insights with convolutional neuronal nets in the near future and are glad to contribute along this way.

References

[1] Boecking, Otto and Spivak, Marla. Behavioral defenses of honey bees against varroa jacobsoni oud. *Apidologie*, 30(3):141–158, 1999.

[2] K. Bozek, L. Hebert, A. S. Mikheyev, and G. J. Stephens. Towards dense object tracking in a 2d honeybee hive. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4185–4193, 2018.

[3] Z. Chen, V. Badrinarayanan, C. Lee, and A. Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. *CoRR*, abs/1711.02257, 2017.

[4] D. L. Cox-Foster, S. Conlan, E. C. Holmes, G. Palacios, J. D. Evans, N. A. Moran, P.-L. Quan, T. Briese, M. Hornig, D. M. Geiser, V. Martinson, D. vanEngelsdorp, A. L. Kalkstein, A. Drysdale, J. Hui, J. Zhai, L. Cui, S. K. Hutchison, J. F. Simons, M. Egholm, J. S. Pettis, and W. I. Lipkin. A metagenomic survey of microbes in honey bee colony collapse disorder. *Science*, 318(5848):283–287, 2007.

[5] A. Dutta, A. Gupta, and A. Zissermann. VGG image annotator (VIA). http://www.robots.ox.ac.uk/ vgg/software/via/, 2016. Version: 2.0.6, Accessed: 01.11.2018.

[6] L. Fortel, M. Henry, L. Guilbaud, A. L. Guirao, M. Kuhlmann, H. Mouret, O. Rollin, and B. E. Vaissière. Decreasing abundance, increasing diversity and changing structure of the wild bee community (hymenoptera: Anthophila) along an urbanization gradient. 9(8), Aug. 2014.

[7] W. Ge and Y. Yu. Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning.

In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[8] T. Gebru, J. Hoffman, and L. Fei-Fei. Fine-grained recognition in the wild: A multi-task domain adaptation approach. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1349–1358, 2017.

[9] T. Gebru, J. Hoffman, and L. Fei-Fei. Fine-grained recognition in the wild: A multi-task domain adaptation approach. *CoRR*, abs/1709.02476, 2017.

[10] C. A. Hallmann, M. Sorg, E. Jongejans, H. Siepel, N. Hofland, H. Schwan, W. Stenmans, A. Müller, H. Sumser, T. Hörren, et al. More than 75 percent decline over 27 years in total flying insect biomass in protected areas. *PloS one*, 12(10):e0185809, 2017.

[11] L. Hein. The economic value of the pollination service, a review across scales. *The Open Ecology Journal*, 2(1):74–82, Sept. 2009.

[12] M. Henry, M. Beguin, F. Requier, O. Rollin, J.-F. Odoux, P. Aupinel, J. Aptel, S. Tchamitchian, and A. Decourtye. A common pesticide decreases foraging success and survival in honey bees. *Science*, 336(6079):348–350, 2012.

[13] G. V. Horn, O. Mac Aodha, Y. Song, A. Shepard, H. Adam, P. Perona, and S. J. Belongie. The inaturalist challenge 2017 dataset. *CoRR*, abs/1707.06642, 2017.

[14] C. M. Kennedy and E. Lonsdorf. A global quantitative synthesis of local and landscape effects on wild bee pollinators in agroecosystems. *Ecology Letters*, 16(5):584–599, Mar. 2013.

[15] H. Li, K. Ota, and M. Dong. Learning iot in edge: Deep learning for the internet of things with edge computing. *IEEE Network*, 32(1):96–101, Jan 2018.

[16] W. Liu, D. Anguelov, D. E. Erhan, C. Szegedy, S. Reed, C. Fu, and A. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[17] F. Nazzi and F. Pennacchio. Honey bee antiviral immune barriers as affected by multiple stress factors: A novel paradigm to interpret colony health decline and collapse. *Viruses*, 10(4), 2018.

[18] R. J. Paxton. Does infection by nosema ceranae cause colony collapse disorder in honey bees (apis mellifera)? *Journal of Apicultural Research*, 49(1):80–84, 2010.

[19] T. D. Pereira, D. E. Aldarondo, L. Willmore, M. Kislin, S. S.-H. Wang, M. Murthy, and J. W. Shaevitz. Fast animal pose estimation using deep neural networks. *bioRxiv*, page 331181, 2018.

[20] S. G. Potts, J. C. Biesmeijer, C. Kremen, P. Neumann, O. Schweiger, and W. E. Kunin. Global pollinator declines: trends, impacts and drivers. *Trends in Ecology & Evolution*, 25(6):345 – 353, 2010.

[21] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley. Color transfer between images. *IEEE Computer Graphics and Applications*, 21(5):34–41, 2001.

[22] I. Rodríguez, K. Branson, E. Acuña, J. Agosto-Rivera, T. Giray, and R. Mégret. Honeybee detection and pose estimation using convolutional neural networks. Technical report, Technical report, RFIAP, 2018.

[23] I. F. Rodriguez, R. Megret, E. Acuna, J. L. Agosto-Rivera, and T. Giray. Recognition of pollen-bearing bees from video using convolutional neural network. `https://github.com/piperod/PollenDataset`, 2017.

[24] I. F. Rodriguez, R. Megret, E. Acuna, J. L. Agosto-Rivera, and T. Giray. Recognition of pollen-bearing bees from video using convolutional neural network. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, volume 00, pages 314–322, Mar 2018.

[25] A. Rosebrock. Super fast color transfer between images. `https://github.com/jrosebr1/color_transfer`, 2004.

[26] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018.

[27] C. W. Schneider, J. Tautz, B. Grnewald, and S. Fuchs. Rfid tracking of sublethal effects of two neonicotinoid insecticides on the foraging behavior of apis mellifera. *PLOS ONE*, 7(1):1–9, 01 2012.

[28] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, Oct 2016.

[29] S. L. Smith, P. Kindermans, and Q. V. Le. Don't decay the learning rate, increase the batch size. *CoRR*, abs/1711.00489, 2017.

[30] J. Sung. Ssd in tensorflow: Traffic sign detection and classification. `https://github.com/georgesung/ssd_tensorflow_traffic_sign_detection`, 2017.

[31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[32] M. Teichmann, M. Weber, M. Zllner, R. Cipolla, and R. Urtasun. Multinet: Real-time joint semantic reasoning for autonomous driving. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1013–1020, June 2018.

[33] D. vanEngelsdorp, J. D. Evans, C. Saegerman, C. Mullin, E. Haubruge, B. K. Nguyen, M. Frazier, J. Frazier, D. Cox-Foster, Y. Chen, R. Underwood, D. R. Tarpy, and J. S. Pettis. Colony collapse disorder: A descriptive study. *PLOS ONE*, 4(8):1–17, 08 2009.

[34] Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003.

[35] F. Wario, B. Wild, R. Rojas, and T. Landgraf. Automatic detection and decoding of honey bee waggle dances. *PloS one*, 12(12):e0188626, 2017.

[36] M. E. Watanabe. Colony collapse disorder: many suspects, no smoking gun. *Bioscience*, 58(5):384–388, 2008.

[37] B. Wild, L. Sixt, and T. Landgraf. Automatic localization and decoding of honeybee markers using deep convolutional neural networks. *CoRR*, abs/1802.04557, 2018.