Freie Universität Berlin

Master's Thesis at the Department for Informatics and Mathematics

Dahlem Center for Machine Learning and Robotics - Biorobotics Lab

Classification of Honeybee Larval Stages Using CNNs Applied to Image Data

Adrian Defèr
Matriculation Number: 4726300
a.defer@fu-berlin.de

Supervisor: MSc. David Dormagen
First Examiner: Prof. Dr. Tim Landgraf

Berlin, January 28, 2022

Abstract

The decline of honey bees (Apis mellifera), due to various threats, like the ongoing global climate change or pesticides, endangers wild plant diversity, ecosystem stability and crop production. The EU funded project *Hiveopolis* wants to address this problem with technology. A newly developed intelligent bee colony system equipped with sensors, actuators, and robots will be used to optimally manage and guide the bee colony through nowadays challenges. Part of the research within the *Hiveopolis* project deals with automated methods for monitoring the brood nest on a honeycomb. Which is useful for assessing the colony strength. This thesis leverages high resolution image data of a honey bee colony, recorded with the hive observation setup, from the *BeesBook* project at the Biorobotic Lab, whose team also contributes to *Hiveopolis*, at the Freie Universität Berlin, in order to investigate how well it is possible to predict honey bee brood age with high resolution image data. The developed deep learning system serves as a baseline for a reference system, which will provide ground truth data for teaching more inexpensive, non-invasive, and space-saving machine learning systems, such as temperature sensor based systems, with the sensors installed on the back of the honeycomb. Since, with a prediction error of 2.1 ± 26.5 hours (mean $\pm$ standard deviation) and a mean absolute error of 0.66 days, the system does not operate at reference level. It is found, that even if the image data is highly resolving, the system would benefit from cameras whose focus is set on the interior of the cells and from improved lighting of the inside of the cells. But also from human-performed quality control of the automatically generated training data set.

Eidesstattliche Erklärung

Ich versichere hiermit an Eides Statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

28. Januar 2022

Adrian Defèr

Contents

List of Figures

1 Introduction

The first part *Motivation* provides a short introduction into the threats to which honey bees are exposed these days and how these threats could be largely mitigated in the future through a novel project. The second part *Outline of Contribution* describes how Deep Learning methods are used to asses the health of a honey bee colony. The third part *Structure of this Thesis* gives an insight into how this work is structured.

1.1 Motivation

Honey bees (Apis mellifera) fulfill an important function in the pollination of plants. Due to their medium body size, 10 to 15mm long[9], and medium-long proboscis – about 6mm when extended[40, 10] – honey bees can pollinate a wide variety of flowers. Besides being a key component of global biodiversity and providing ecosystem services to crops and wild plants they are of enormous value for the industrial food supply[15]. 90% of commercial pollination services are provided by managed honey bees[16]. But native pollinators, and also managed honey bees, are in decline. Which has been the subject of research for more than a decade now[31, 24]. Threats to honey bee colonies like the ongoing global climate change[44], pesticides[44, 8, 49, 35], viruses[16, 32], bacteria[16] and fungi[16] endanger the fulfillment of this biological and economical function of pollination and therefore the maintenance of wild plant diversity, ecosystem stability and crop production.

The EU funded project Hiveopolis[1] aims to create a blueprint for a hybrid of bee colony and technology that is equipped to meet the challenges of today. With the help of sensors, actuators and robots – embedded in the honeycomb – the bee colony should be optimally controlled. The guidance of the bees, through actuators and robots, should in turn be controlled by expert knowledge embedded in algorithms, the evaluation of the installed sensors and external data, such as weather forecasts and publicly accessible databases. For example, a smart honeycomb will be equipped with a dancing robot to guide collecting bees to specific food sources or vibration plates to prevent bees from seeking out harmful food sources. Or as a result of predicted prolonged bad weather, bees could be encouraged to build up stores of pollen.

At the Biorobotics Lab[2], led by Prof. Dr. Tim Landgraf, at the Dahlem Center For Machine Learning Robotics[3], workgroup Artificial & Collective Intelligence, at the FU Berlin, artificial and collective intelligence is being researched using bees and fish as models. Due to their medium small body size, small colony footprint, relatively short life cycle and unique social behavior within a superorganism, honey bees are a popular subject of research. The ultimate goal of the Biorobotics Lab, in terms of honey bees, is to entirely understand their social structures and how the collective intelligence is created based on the predetermined life paths of the individual bees.
From 2019 on the Biorobotics Lab contributes to the Hiveopolis project. One of these contributions is the aforementioned dancing robot[23].

Part of the research within the Hiveopolis project also deals with automated methods

[1] https://www.hiveopolis.eu/ [Online; accessed January 04, 2022]
[2] https://bioroboticslab.github.io/website/ [Online; accessed January 04, 2022]
[3] https://www.mi.fu-berlin.de/inf/groups/ag-ki/index.html [Online; accessed January 04, 2022]

for brood nest monitoring. This is because it is possible to infer the fitness of the queen bee and the condition of the colony as a whole from the status of the brood nest – a specific area on the honeycomb or several combs dedicated for brood in larger colonies. Gaps in the brood nest, for example, indicate a weakened queen. So does the laying of eggs in a pattern other than the typical spiral pattern[4]. If brood is eaten by other adult bees, this may indicate a pollen shortage. If brood does not develop to the end, for example due to starvation or freezing, this can mean that the colony has shrunk too small. So that the remaining bees can no longer fulfill the tasks of feeding and warming the brood. Thus, the shape of the brood nest, the brood density and the brood survival rate can already be good metrics for the health of the entire bee colony. The implementations of all functions should, of course, be as cost-efficient as they are space-saving. A non-invasive, space-saving and rather inexpensive method of monitoring brood nests could therefore be implemented using temperature sensors on the back of the honeycomb. A temperature of 34.5°C to 36.0°C is considered to be optimal for rearing bee brood[43]. The worker bees use their wing muscles and targeted vibrations with their bodies in the cells around the brood nest to ensure that the brood constantly maintains this temperature. And the larger the brood, the more it heats the cell itself.

Thus, one could try to infer the brood content of a cell from its temperature profile. But which temperature profile corresponds to which brooding state? This could be learned from a machine learning model fed with appropriate ground truth data. This ground truth data could, in turn, be generated with a one-time experimental setup using the temperature sensors on the back of the comb and a camera based system that covers the front of the comb at the same time over a certain period. The evaluated image or video data from the camera system, could then yield the target values for the input-target pairs, e.g. a temperature curve for a cell over a certain time period until time t (input) and the corresponding brood age at time t for that cell (target), which are needed to teach the temperature sensor system how to derive the brood age from a temperature profile. However, this procedure only needs to be done once for a particular setup. Since in practice the exact design of such a system will vary, e.g. the thickness of the back wall of the comb, the exact hardware characteristics or placement of the sensors, it would be advantageous if the aforementioned evaluation of the data from the camera system were not performed manually, but automatically by an other machine learning system, that is robust to slightly different shaped combs and brood cells, as well as different lighting or image resolution. An experimental setup that is well suited for developing exactly such a system exists at the Biorobotic Lab, see figure 3.

1.2 Outline of Contribution

The aim of this thesis is to provide a baseline on the prediction of brood ages based on high resolution image data, recorded with the experiment setup at the Biorobotics Lab in the season of 2019. Before attempting to solve this task under suboptimal conditions – different sensors, lower resolution, poorer lighting – this work will examine how well it is actually possible to predict brood age under optimal conditions.

[4]Honey bee queens lay eggs in a spiral pattern, starting from the center of the comb and laying one egg per cell, from cell to cell. Leaving no single cells empty within the brood area. This is considered to be the most efficient way – both in terms of the process of laying and from the energy efficiency for incubation[36].

For this purpose two machine learning systems were developed in this work. The first one solves the problem of obtaining image coordinates of cells, whose interior is not obscured (usually from a bee walking over the comb or a wax cap), from an image of the whole comb (see figure 4). The second model predicts the age of a honey bee brood from an image of a honey comb cell. The two systems together could be used to scan the surface of a honeycomb for bee brood and track its development over time.

1.3 Structure

The following section *Background* consists of four parts. The first part *Stages of Development of the Honey Bee* provides the necessary knowledge about honeybee development relevant to this work.
The second part *Recording Setup* describes the hardware setup of the recording season of 2019 at the Biorobotics Lab and gives some notion of the footage recorded with it.
An overview of the machine learning methods applied to the data set, which is extracted from the raw footage is given in the third part *Artificial Neural Networks in Computer Vision*.
The last part *Related Work* outlines the relation of this thesis to other scientific work and the current state of research concerning the classification of honey bee brood age on image data.

The third section *Non Covered Cell Localizer* describes how the model for the detection of non covered cells on the comb is implemented and evaluates its performance.

The fourth section *Larval Age Classifier* describes how the model for the prediction of the brood age on image patches is implemented. An evaluation of the performance of this model is given.

The fifth section *Results* provides the results of the trained models presented in section three and four.

The last section *Conclusion* discusses the final results, assesses both the potential benefits in practice and the limitations and gives an outlook on how the results can be improved through future work.

2 Background

The first part *Stages of Development of the Honey Bee* gives background to the life cycle of honey bees and elaborates on the specific stages in the development of a honey bee.
The second part *Recording Setup* gives a description of the hardware setup of the recording season of 2019 at the Landgraf Lab and gives a basic impression of the properties of the dataset that was worked with in this thesis. The third part *Artificial Neural Networks in Computer Vision* introduces convolutional neural networks. In the last part *Related Work* seven papers and completed theses, which relate to this work in salient aspects, are presented and their main statements are elaborated. The current state of research concerning the classification of honey bee brood age on image data is provided.

2.1 Stages of Development of the Honey Bee

Figure 1 depicts the development of a honey bee, specifically a worker bee. While all bees share the same developmental stages, the time spend in each stage is different for

each of the castes – worker bees, drones and queens. According to [50] the development of the european honey bee is considered to take 21 days on average, spending 3 days as egg, 9 days as larva (of which the last two days it is called prepupa) and 9 days as pupa.

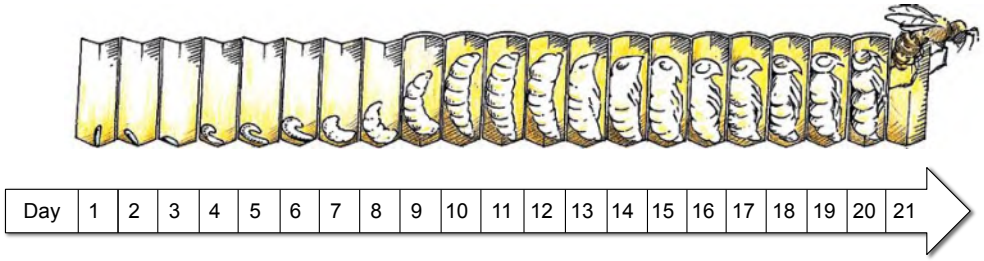


Figure 1: The Development of a Worker Bee. The 4 major stages of development are: egg (day 1-3), larva (day 4-12), pupa (day 13-21), adult (after emerging from the capped cell, around day 21). The queen bee lays single fertilized eggs into dedicated worker cells. After 3 days the embryo hatches from the egg. Now the larval stage begins. The larval stage consists of an uncapped and a capped phase. The former is the feeding time, in which the larva gains about five hundred times its weight. The cell of the larva is capped with a wax lid after 9 days by adult workers. Now the larva uncurls, stretches out fully, spins its cocoon inside the cell and enters the pupal stage around day 13. Now the metamorphosis begins. Once, its transformation is completed, around day 21, it bites its way through the wax capping and leaves the cell as an adult. The growth of the bee is now terminated.[50] Reproduced from [21].

The queen bee lays eggs, one per cell, into the comb by gluing them on the floor of the cells, that are dedicated for rearing worker bees – usually in the center of the comb. That is why they appear to be standing on the floor of the cell. Which can bee seen on the right half of the image in figure 2. Unfertilized eggs will develop into drones, the fertilized ones will, depending on the nutrition, develop into worker bees or queens. After three days, the egg has been sagged to the ground during this time, the embryo hatches from the egg, which is more of a dissolution of the eggshell over time. Now the larval stage begins. The first, uncapped, phase of this stage – the open larval stage – is the feeding time, in which the larva gains about five hundred times its weight. At the beginning all larvae are fed with royal jelly, which is a glandular secretion of nurse bees[5]. It is this milky white secretion, in which the larvae lie in the upper left half of the image in figure 2. It consists mainly of water and contains proteins, sugars and lipids. It is generally accepted in research that larvae develop into queens when fed exclusively with royal jelly. Larvae, which in contrast are to be reared to worker bees or drones, are fed with a mixture of bee bread (fermented pollen), honey and secretion after day three, the so called worker jelly[29]. The cell of the larva, which is now a thick, round maggot is capped with wax after nine days by nurse bees – the capped larval stage begins. Now the larva uncurls itself and stretches out fully, with its head part towards the capped end of the cell. Then the larva enters the prepupal stage and spins its cocoon inside the cell. Around day thirteen it enters the pupal stage, in which, via metamorphosis, the

[5]Nurse bee is the first of several jobs that the worker bee, after cleaning its brood cell and thus making it ready for the next egg, will perform during its few weeks of life.

pupa turns into an adult bee. Once this transformation is complete the adult worker bee bites its way through the cell capping and leaves the cell. The growth of the bee is now terminated. During the next few days the development will complete. Her hair will dry, the shell will harden and the stinger will become functional.[50]

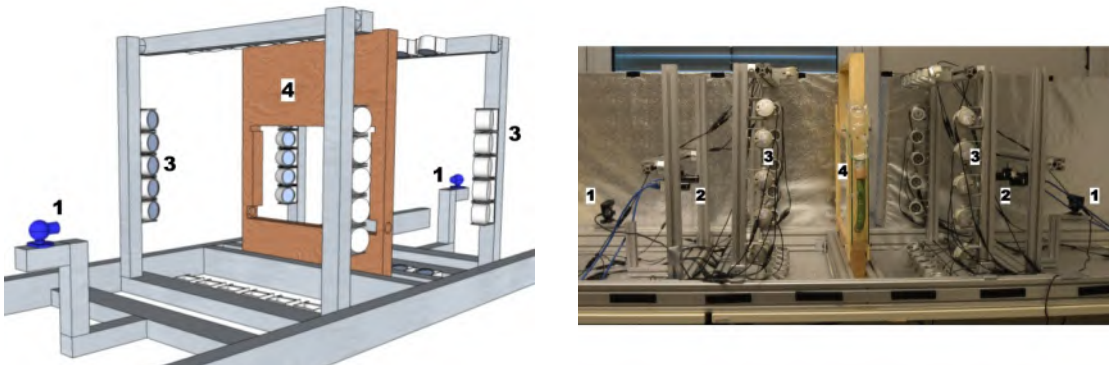
The developmental period of interest for this work extends from the egg to the open larval stage. Only in these stages it is at all theoretically possible to determine the age of the brood, based on its optical characteristics. And even here it becomes very difficult for the prediction of the correct egg stage, since the egg does not really change in size. According to [50] it loses about 30% of its weight. Maybe that translates to a decrease in size. But it is questionable whether such a difference, with an average egg size of 1.3 to 1.8mm, can be recognized at all on the image data. In addition, the egg size can vary greatly even for the same queen bee. The only chance seems to be to derive the age from the angle relative to the cell walls. Since The egg gradually sags and rests on the floor at the end of the three day period[50]. With a cell capped, there is no chance for directly predicting the brood age. Then one only has the chance to deduce it from the context, the surrounding cells. Therefore, this work is limited to the analysis of the image data from the first two stages – egg and open larval stage. Which in principle is sufficient. Since once a certain period of time has been classified correctly for a given cell, its age can be extrapolated based on that. Assuming the average total development time.

In the following, unless explicitly stated otherwise, the term bee always refers to an individual of the genus Apis mellifera – the western honey bee or european honey bee.

Figure 2: Photo of Eggs and Larvae in the Brood Nest. This photo shows eggs and larvae of honey bees in cells of a comb, whose side walls have been removed. In this case, according to the author, it is drones. The photo shows the larvae about 3 to 4 days after hatching, hence the total age of the larvae on this photo is about 6 to 7 days. They lie in a milky white secretion on which they feed. The eggs stand upright on the bottom of the cells. Reproduced from [46].

2.2 Recording Setup

In order to be able to study the life and social structures of honey bees, as part of the project 'BeesBook', a honey bee colony is housed directly at the institute. For each recording season, a bee colony is moved from its breeding site to a beehive at the institute. Individuals are then selected from this 'stock' hive and placed in the observation hive. The prepared observation hive, depict in figure 3 consists of a prefabricated, double sided honeycomb attached to a wooden frame. This observation hive is installed in a darkened room in such a way that the bees can be observed with cameras over their entire life span. Both sides are covered with glass and sealed so that there are only two access points to the interior. One towards the room, to which a plastic tube is connected, designed as a sluice for adding new bees. The other one on the opposite side, which, through a hole in the window, leads outside and serves as an entrance as well as exit for the bees.



(a) (b)

Figure 3: Recording Setup. **(a)** A CAD model of the recording setup. **(b)** A photo of the recording setup. Both images depict the recording setup from 2016 and show: **1** one PlayStation-Eye low resolution, high fps camera on each side with removed IR (infrared) filters, for waggle dance recording[45]. **2** Two high resolution, low fps cameras on each side, covering a half of the comb each, for tracking individual bees (i. e. decoding the circular tags of the bees[37, 47] and record their trajectories based on that[4, 51, 5]). **3** IR light ($840nm$) arrays, for illumination invisible to bees, **4** Wooden frame with glass covers, this is where the honeycomb is hung. For the current setup the illumination was updated and the high resolution cameras have been replaced with a single one on each side, yielding 12MP at 6fps. Adapted with permission from [1].

Before put into the observation hive, each bee is marked with a circular binary code on their thorax[37]. This allows for the fully automatic identification[47] and tracking[6] of individual bees via computer vision over their entire lifespan. All experiments are related to the social behavior of worker bees, since drones, except for fertilizing the queen, do not contribute much and are more of a burden to the worker bees. To ensure that only marked worker bees are in the observation hive, only worker bees, and initially the queen, are selected from the 'stock' hive and placed, with a marker attached, in the observation hive. In addition to that the combs get replaced every eighteenth day. Which ensures that no drones or unmarked worker bees hatch from the cells. New hatched worker bees get added to the observation hive every day. In the season of 2019, a total of about 6000 bees were placed in the observation hive.

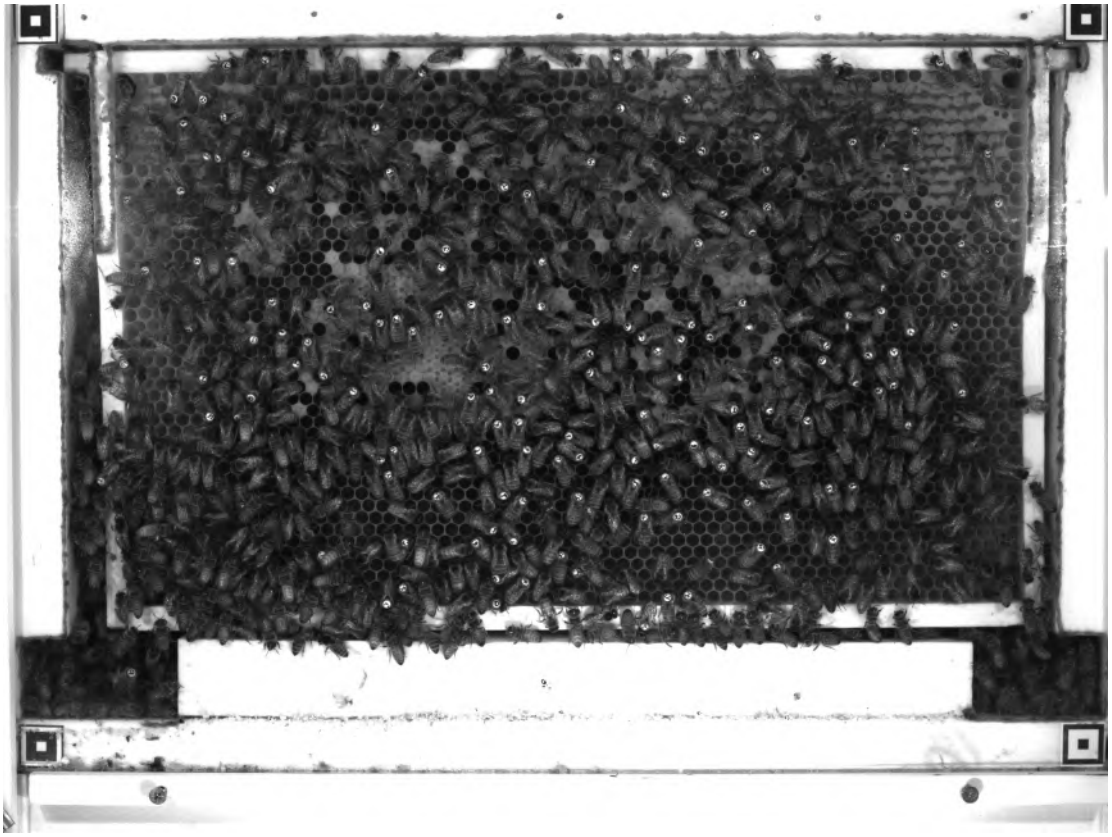


Figure 4: Sample of the Footage from Recording Season 2019. This image shows the comb surface at 15:28:46 on 2019-08-01, shot with *Cam_0* – one of the 12MP cameras recording the observation hive. It has a resolution of 3000x4000px in height and width and is only grayscale. The capped cells in the upper left and upper right corner of the comb are filled with honey and serve as stores. The capped cells in the center, between the busy bees, are the cells that contain a pupa. These are the kind of images on which this work is based.

The observation hive on each side is filmed through the glass panes by a 12MP camera, providing 6fps – sufficient resolution for decoding the markers – along with a low-resolution camera at 60fps. The latter yields enough frames per second to analyze the bees waggle dances[45]. The data stream of this high frame rate camera is not saved by default. But only when a waggle dance is detected. Figure 4 shows example footage taken with the former high resolution camera. The comb gets illuminated indirectly with infrared flashes, synchronized with the high resolution cameras, in order to minimize reflections on the glass panes. That is why the infrared lights, see figure 3, item 3, are arranged parallel to the comb and not directed at it. The observation hive gets fully covered by reflective foil, like the one to be seen in the background of figure 3 (b). Which provides a good indirect lighting of the comb surface. In order to mitigate back illuminated images of the comb, the images are taken alternately on each side. Additionally the surface of the combs gets illuminated with red light constantly in the area of the so called dance floor, on which the forager bees perform their waggle dance to recruit their colleagues. Usually the dance floor extends over an area near the entrance. Compared to humans, the spectrum of light visible to bees is shifted by about $150nm$

toward the shorter wavelength spectrum. Bees can therefore perceive ultra violet light, but not red light, which makes the whole lighting invisible to them.

2.3 Artificial Neural Networks in Computer Vision

Algorithms that mimic human thought, action and reasoning at the same time, such that they embody all the characteristics of human intelligence, are referred to as artificial intelligence (AI). One subset of AI is machine learning (ML). ML algorithms nowadays can accomplish individual small subtasks of a general AI. Such as classifying an image into a number of predefined classes or what one should watch next on ones favorite streaming platform. Not what one should do next in general. But if one would be in the mood for a new series tonight, which would be the one most likely to be enjoyed. So these algorithms are limited to specific tasks. In principle, these algorithms learn by imitation. They learn from examples that are presented to them, along with the desired output for that example. Since the algorithm generally cannot be shown all possible examples that may occur in reality, the algorithm ideally induces general rules for solving its problem. Instead of just memorizing the correct solution for all examples. Which leads to unpredictable results for new, previously unseen examples. ML spans many methods. Artificial neural networks (ANNs)[33] are the class of algorithms, that boosted machine learning the most, in recent years. Vanilla (multi layer) feed-forward ANNs are made up of several units called neurons, as they are biologically inspired, which are arranged in layers. Each unit has its own assigned so called weight, a scalar which is learnable. Each unit receives the outputs of the units in the previous layer, aggregates them and multiplies it by its weight value. After applying a non-linear function to this weighted aggregation it passes the result to the neurons in the next layer. The units have no connection to the other neurons in the layer to which they themselves belong. ANNs can therefore be understood as a big function composition, which can theoretically approximate any imaginable target function due to the non-linearities, if equipped with a sufficient number of neurons. In traditional pattern recognition methods, feature extraction was done by hand and a classifier was trained based on the output of these handcrafted feature extractors. ANNs are designed to be applied to raw data and learn the relevant features from it by themselves.

A special flavour of artificial neural networks are convolutional neural networks (CNNs), first successfully applied in practice by Yann LeCun[27, 26]. Due to their local connectivity and weight sharing they are much more efficient in the application on image data, where they can benefit from learning translation invariant features.

In 2012 AlexNet[22] outperformed state of the art architectures on the ImageNet[12] Large Scale Visual Recognition Challenge 2010[6] by a wide margin, using a deep (5 convolutional, 3 dense layer), ReLu[30] activated CNN architecture, together with max-pooling, dropout[39] and image augmentation, trained on standard end-user graphics cards. What was downright revolutionary at that time. With this work Alex Krizhevsky ensured deep CNN's widespread use in image classification. To this day, CNNs are still state of the art in computer vision.

Another milestone in the history of deep neural networks is the introduction of skip connections, which feed the output of some layer l not just into the next layer $l+1$ but also into layer $l+2$ or $l+3$. Which mainly counteracts the problem of the vanishing

[6] https://www.image-net.org/challenges/LSVRC/ [Online; accessed January 13, 2022]

gradient[18] and thus allows for the training of not just deep but very deep neural networks. The architecture applying this principle is the so called ResNet[17] (residual neural network) architecture.

In this work a fully convolutional ResNet architecture with fixup initialization[52] is used for both tasks – the localization of non covered cells and the classification of the brood inside a cell – with slight adjustments in each case. Fixup initialization rescales the initial weights of each layer according to its depth, making the network better conditioned for backpropagation[25] and thus eliminates the need for Batch Normalization[20]. Which takes away some computational load in the training process. The deep learning models are implemented using PyTorch[7].

2.4 Related Work

In [42] the effect of pollen shortage on larval survival rate is investigated, since pollen is the most important protein source for bees. In addition, it was investigated whether this influence also has an effect on the total time span of egg and open larval stage, i. e. the time span from the laying of the egg to the capping of the cell. To do this, the authors, on the one hand, compared foraging with non-foraging periods and, on the other hand, periods with manual pollen reduction with periods without pollen reduction. Non-foraging periods were simulated with a rain machine at the entrance of the hive, preventing the bees from leaving the hive. It was assessed by the authors, that the rain was accepted by the bees as natural. Which is crucial, as a simple blocking of the entrance e. g. would, according to the authors, result in an intracolonial chaos. Manual pollen reduction consisted of a pollen trap at the entrance and regular replacement of pollen-filled combs with empty combs. In the experiment on pollen reduction the bees were not prevented from leaving the hive in any way. This was done to exclude side effects introduced by e. g. the lowered temperature due to the rain machine or the simultaneous scarcity of nectar. Furthermore it could be tested, whether the effects of non-foraging periods were mainly due to the shortage of pollen to which the bees were exposed. And this was exactly the case. It was found that a bad pollen supply led to cannibalism of the younger larvae. In contrast it had almost no influence on older larvae at all. Older larvae were capped earlier than normal, which reduced their demand for protein. At the same time the cannibalism of younger larvae enriches the worker jelly with protein. In a nutshell, the bees save the larvae in which they have already put most of their investment and which will soon no longer need investment.
In order to track the state of the brood nest in each of their experiments the authors mapped the whole hive on transparent sheets against the glass wall of the observation hive. Thereby classifying the developmental stage of each individual. The exact age was known, due to the fact, that the egg laying by the queen happened under precise, limited conditions. After egg laying, within a window of two hours, the queen was excluded from the brood nest. Since no exact procedure is mentioned, the author assumes that the measurements and calculations were all done by hand.

A very similar experiment in [11], by the same lead author, studied the exact nursing behavior and the behavior of the queen itself, under basically the same circumstances, in more detail. The experiment was conducted with the same rain machine approach as

[7] PyTorch: An Imperative Style, High-Performance Deep Learning Library; https://pytorch.org/ [Online; accessed 25 January 2022]

in [42]. Additionally they observed four selected larvae twice a day under red light for thirty minutes each and recorded all nursing acts, dedicated to them. They found, that the impact on the nursing of young larvae decreased to a small fraction of the initial level. Even until four days after a non-foraging period. In contrast nursing periods for older larvae were reduced too. But only until the third day. Then they increased again, regardless of whether the non-foraging period was already over or not. They concluded that, as suspected in [42], under prolonged non-foraging conditions, the nursing of young larvae declines. Together with cannibalism by the nursing but also the queen bee the larval population declines. Then, most nursing efforts are put into the surviving older larvae. As these represent a greater investment of resources, as stated in [42].
Again the tracking of the brood nest, as well as the state of pollen and nectar cells, was done by mapping all the cell contents onto transparency sheets every day. Again assumed to be evaluated all by hand.

In [38], the author claims to have presented the first ever sociometry of honey bees. In its own biological context, a sociometry is an all-encompassing account of the social life of insects. A record of all physical and numerical attributes, that helps us understanding how insect colonies develop. Which are for instance the size of the population, with drones and worker bees considered separately, the comb area, the comb use, the swarming rate or the colony death – when and how they die. Although several attempts have been made, every single one, according to the author, lacks of some crucial aspect to a comprehensive sociometry. For this reason, in [38] four separate colonies were observed for about a year and a half, from the beginning until their death. Observing the brood nest was, of course, one of the tasks that had to be accomplished.
The measurements of the content of the combs was performed once a week. Like in [42] and [11] the content was recorded by tracing the area of brood (in general), pollen, honey or empty cells on transparency sheets. Here, however, the sheets were subsequently photographed together with a size reference and then measured digitally, using a common image editing program. The author himself admits that this method had its inaccuracies, due to the tracing technique. And of course due to time constrains. If done manually, there is only limited time for recording the state of a comb. And if not everything is exactly visible at that time, then the rest has to be estimated.

In [48] video recordings from 25 consecutive days of a whole colony, conducted similarly to [6] and as depicted in figure 3, are evaluated extensively. By continuously decoding the markers on the bees trajectories of all individuals of the whole colony are extracted. Together with the division of the comb into specific areas, which are associated with the performance of specific tasks, the social network structure of the entire colony is determined. A simple single descriptor 'network age' is then introduced, which expresses a bee's social network for a specific day. This encompasses e. g. the social interactions and the place of it in the overall social structure of the colony. It is found that the network age can better predict future tasks, social interactions and even the bee's death than the actual biological age of it.
In order to determine where, semantically, a bee was at a given time, these areas, like dance floor or brood area, had to be defined beforehand and updated every day. These tasks partly were performed manually, but at least in digital form. Background images were extracted from the video footage by applying a rolling median filter to a subset of the images of a day and then selecting the modal pixel value of all these median filtered images as the corresponding pixel in the background image. In these

background images, the authors manually outlined the capped brood area for every day. Afterwards, to obtain the open brood area – cells containing eggs or larvae – the area of the comb that would become capped within 8 days was calculated.
Therefore, the video recording of the experiment eliminated the task of manually tracing the area and digitizing the result afterwards. The temporal resolution of a whole day is quite sufficient for this experiment. And the purely digital method is far more accurate in spacial terms, compared to the method described in [38]. However, an automated method that can detect the brood cells would have eliminated the last bit of manual work for that specific task.

An advanced solution to provide automated cell detection, released as open source[8], is presented in [2]. Using the Circle Hough Transform[13] (CHT) implementation from OpenCV[9] cells are localized and subsequently cleaned up from false positives via semantic segmentation of the comb as a whole leveraging a U-Net[34] architecture network. Image patches of the localized cells are then classified by a MobileNet[19] CNN model. Ten different architectures are evaluated. The MobileNet architecture represents the best compromise between training time, inference speed and classification accuracy, with a F1 score of 0.94 over all 7 classes – egg, larva, pupa, pollen, nectar, honey, empty/other. The performance of the localization of the cells is given with a Cell Detection Rate, defined by the authors as in Eq.1, of 98.7%. Where Detection Count is the number of localized cells by the CHT, FP are the false positives determined by segmentation of the input image – which simply means to throw away all detections that occur in the image background – and Manual Count is the number of manually counted cells on the respective image, disregarding the actual position of the cells – if understood correctly.

$$\text{Cell Detection Rate} = \frac{\text{Detection Count} - \text{FP}}{\text{Manual Count}} \times 100\% \tag{1}$$

What distinguishes the work described from this one is its application to color images. Which means the classifier can draw on more information than with gray scale images as depict in 4. Furthermore it is applied to images of combs removed from the hive, shot by any users of this software. Which on the one hand makes it easier due to the fact, that there are no obscuring bees. But on the other hand the algorithm has to be more robust to variation in image size, i. e. to the size of the cells in pixels, lighting, shooting angle, image distortions or image noise, due to different people shooting with different cameras. Using the same camera, shot from the same angle, with fairly consistent lighting and same exposure, makes it much easier in this regard. Last but not least, in contrast to this work the work described classifies the cells localized into their types, regarding content. Brood is thereby classified only according to it's stage and not according to it's exact age.

In [7] a system for markerless tracking of an entire honey bee colony is presented. The authors achieved to recover about 79% of manually labeled bee trajectories from different observation hives over a time span of 5 minutes. Instead of decoding markers they exploit distinct visual features of the bees in order to match identical bees in different video frames to reproduce trajectories from. The authors, like [2], use the U-Net architecture to segment the honeycomb in order to locate bees. Rather as a byproduct, they

[8] DeepBee© https://100prs.dev/DeepBee [Online; accessed 21 January 2022]
[9] OpenCV – an open source computer vision and machine learning software library, https://opencv.org/ [Online; accessed January 21, 2022]

also trained a model for the detection of capped brood cells[10] on background images, generated from images spanning a time of 12 hours. For this purpose a U-Net model was trained to binary classify all pixels of a into either capped brood cell or something else. The labeling of the area of the brood cells leaves out the cell borders, which allows for the localization of single brood cells in post-processing steps, after inference.
The limitations in contrast to the goals of this work are, that the detection is only done on background images and not on real time video footage of a honeycomb in operation. Furthermore classification is only done into capped brood cell or not. Ignoring everything else that is done in terms of bee tracking.

A similar result as in [7] was achieved in [14]. The subject of this thesis was the detection of capped brood cells on background images, like the ones generated in [48], but over smaller time spans of 3 hours each. Unfortunately, this work cannot be used to localize single instances of capped brood cells, since it is more of a segmentation of the whole capped brood area. With a model for the exact localization of capped cells, applied to just such median images, it would have been possible to generate any number of training data for the larval age classifier in this work.
Nevertheless, the manually created markers of capped brood cells on 52 background images can be used again in this work, as described in part 4.1 of section *Larval Age Classifier*.

To the best knowledge of the author there currently does not exist any automated method, that is capable of predicting the age of honey bee brood given an image of a brood cell. However, there are automated methods that can detect brood cells in general. All methods for determining the exact age of an egg or a larva, found in literature, rely on knowing the exact time the egg was laid by the queen. In bee research, a lot of manual work is still done to observe the individual cells of honeycombs in detail. Fully automated methods, which could take over this labour-intensive task, would allow experiments to be much larger in scale. Related to the size of a colony or even the number of colonies, as well as the temporal extension of the experiments.

3 Non Covered Cell Localizer

This section describes the implementation of the 'Localizer'. The Localizer fulfills the task of obtaining image coordinates of cells, which are not covered – usually by bees – and therefore the brood in these cells could be seen. The input for this system consists of an image of the whole comb (see figure 4). The Localizer consists essentially of 2 parts. The first one is the deep learning model, whose output can be interpreted as a map of probabilities for a pixel representing the center of an uncovered cell. But with less resolution than the input image. The second part consists of the post-processing, which transforms this probability map into the exact image coordinates of uncovered cells.

[10] The detection of only capped brood cells but not open brood cells too is an assumption made by the author, since the target class is called 'brood cell'. But all visualizations available depict only marked capped brood cells.

3.1 Ground Truth Data

The data basis for the training data are the recordings of the observation hive of season 2019, obtained as described in [6]. 64 images, 32 of each of both cams, were extracted from the video files, distributed over the duration of 10 weeks (2019-07-24 - 2019-10-01). One example is depict in figure 4. In order to teach the model, what objects on the input image it should localize, ground truth data has to be produced. In this case the ground truth data consists of image coordinates of center points of the required, non covered, cells. Hence 64 images were annotated. For this purpose a tool 'CenterClicker' was written. It is a Python script, which loads a given image or all images contained in a given directory and uses OpenCV to display the current image, as well as the currently selected ROI of it. Which is depict in figure 20. The mouse is used to add or remove annotations and select ROIs. Other functionalities are controlled via key bindings. The image coordinates are saved into a CSV file – a universal format that allows to easily import the coordinates by other modules.

Because annotating ground truth data, especially hundreds or thousands of cells, is a labour consuming and tedious task, an object recognition method was used, which automatically annotates the majority of all cells. These annotations were then revised using the CenterClicker, so that cells are annotated which are at least $\sim 75\%$ uncovered. This has drastically reduced the annotation work. More precisely, template matching was used. Unlike with Circle Hough Transform (CHT), as used in [2], see section 2.4, *Related Work* in *Background*, with template matching it can be taken into account that the cells to be detected are open cells. Since, usually, the surface structure of capped cells differs significantly from that of open ones. In addition, more irregularly shaped cells can be detected. And last but not least, there was no need to experiment with the parameters of the CHT.
The procedure was as follows. First, 17 templates, depict in figure 5 were manually extracted from the honeycomb images, covering different image areas – due to image distortions – at different time steps – due to lighting – of both cameras. The only thing that had to be taken care of was that the center of the cell was in the center of the template image. To an input image of the whole comb then template matching is applied

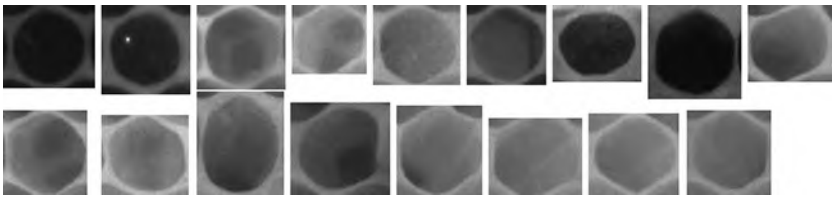
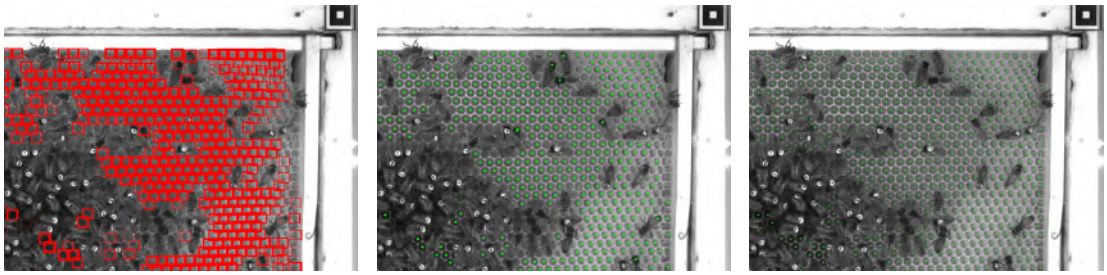


Figure 5: Templates of Open Cells. These 17 templates were extracted from various image frames of both camera's recordings of the observation hive. They were then used for template matching in order to produce ground truth data for the training of the Localizer.

with each template with respective scaling factors of 0.95, 0.975, 1, 1.025, 1.05. The multiple application of one template image with various scalings makes the method more robust to different cell sizes due to lens distortions. Similar to the principle of image pyramids. The OpenCV Python implementation of template matching `matchTemplate` was used. As matching method Normed Correlation Coefficient was used. It was missed to apply Adaptive Histogram Equalization to the templates and input images before-

hand. This would probably have given better results. The image coordinates of the correlation coefficients in the output of `matchTemplate` for all template images, which exceeded a threshold of 0.725, were then clustered using the DBSCAN algorithm (implementation of scikit-learn[11]) with parameters $\epsilon = 5$ and MinPts $= 3$, which is well suited to exclude outliers from the clusters. The geometric means of these clusters were then used as annotations and saved into a CSV file just like the CenterClicker does. This way, a total of 65401 cells were annotated on 64 images. Figure 6 depicts one image of the data set in the 3 stages 'template matching applied', 'clustering applied', 'revision done'.



(a) (b) (c)

Figure 6: Three Steps of Ground Truth Annotation. Crop of one image of the data set in the 3 stages of the annotation process. **(a)** Template matching applied. The red rectangles are bounding boxes of the single template matches. **(b)** Clustering applied. Matches are clustered using DBSCAN. Geometric means of these clusters are used as annotation, depict as green dot. **(c)** Revised output of (b), every annotated cell should be at least 75% visible.

3.2 Model

As mentioned in part 2.3 of section *Background*, the deep learning model of the Localizer, in the following simply referred to as network, uses a fully convolutional ResNet architecture with fixup initialization. Which means, that fully connected layers and Batch Normalization are not used. Hence, the output layer is a convolutional layer, in the following referred to as conv-layer, as well. This enables the application of the network to arbitrarily sized inputs. As long as the input has a certain minimum size. Since this architecture is already successfully used in the current BeesBook setup for localization and decoding of the circular bee tags[37, 47], it was adapted for the localization of non covered cells. Hence a lot of code, with a good chance of success, could be reused and served as a kind of framework.
In total, the network consists of 10 2-D conv-layers, see figure 7, with a kernel size of 3×3 each. 8 of them are embedded in 2 'fixup-layers' of 2 residual blocks each. A residual block consists of 2 conv-layers, of which the output of the second layer is concatenated with the input for the first layer[52, 17]. The 2 fixup-layers are prepended with an initial conv-layer, which reduces image dimensions right at the start, and are followed by a final conv-layer, which reduces the number of channels back to 1. Downsampling is performed by the initial conv-layer and by the first conv-layers of every fixup-layer by applying convolution with a stride of 2. No padding is applied, which is why image

[11] https://scikit-learn.org; [Online; accessed January 23, 2022]

dimensions get reduced by conv-layers even without explicit downsampling. The whole network encompasses 162.000 learnable parameters.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [32, 16, 255, 255]             144
              ReLU-2         [32, 16, 255, 255]               0
            Conv2d-3         [32, 32, 127, 127]           4,608
              ReLU-4         [32, 32, 127, 127]               0
            Conv2d-5         [32, 32, 125, 125]           9,216
              ReLU-6         [32, 32, 125, 125]               0
   FixupBasicBlock-7         [32, 32, 125, 125]               0
            Conv2d-8         [32, 32, 123, 123]           9,216
              ReLU-9         [32, 32, 123, 123]               0
           Conv2d-10         [32, 32, 121, 121]           9,216
             ReLU-11         [32, 32, 121, 121]               0
  FixupBasicBlock-12         [32, 32, 121, 121]               0
           Conv2d-13           [32, 64, 60, 60]          18,432
             ReLU-14           [32, 64, 60, 60]               0
           Conv2d-15           [32, 64, 58, 58]          36,864
             ReLU-16           [32, 64, 58, 58]               0
  FixupBasicBlock-17           [32, 64, 58, 58]               0
           Conv2d-18           [32, 64, 56, 56]          36,864
             ReLU-19           [32, 64, 56, 56]               0
           Conv2d-20           [32, 64, 54, 54]          36,864
             ReLU-21           [32, 64, 54, 54]               0
  FixupBasicBlock-22           [32, 64, 54, 54]               0
           Conv2d-23            [32, 1, 52, 52]             576
================================================================
Total params: 162,000
Trainable params: 162,000
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 32.00
Forward/backward pass size (MB): 2211.31
Params size (MB): 0.62
Estimated Total Size (MB): 2243.93
----------------------------------------------------------------
```

Figure 7: Localizer Model. Output of the function `summary` from the library torchsummary[a]. Shape corresponds to [batch size, channels, height, width]. The network consists of 2 'fixup' layers. Each 'fixup' layer consists of 2 residual blocks (indicated by 'FixupBasicBlock-X', stated after each residual block in summary), which consist of 2 conv-layers each. Input dimensions are [32, 1, 512, 512]. Downsampling in image dimensions is performed by 'Conv2d-1' and by the first conv-layers of every 'fixup' layer, 'Conv2d-3' and 'Conv2d-13' by operating with a stride of 2. Otherwise stride is 1. The last layer 'Conv2d-23' reduces the number of channels back to 1. During training, only the center pixel of the 52×52 output is interpreted as the output of the network. More precisely, the upper left, of the four pixels in the middle.

[a] https://pypi.org/project/torchsummary/ [Online, accessed January 24, 2022]

At training time, only the center pixel of the 52×52 output of the final conv-layer is regarded as output of the network and used for the calculation of the error with respect to the target value. At inference time the network is applied to the 3000×4000 images, like the one depict in figure 4. The output for this input size is 375×500, which is large

enough to image the approximately 44 times 64 cells of a comb individually, see figure 10 (a).

3.3 Data Set

The network is designed to output, for the center pixel in the input image, the probability that this pixel represents the center of an non covered cell. As network input for training, image patches and their corresponding target values have to be generated. From every comb image in the annotated data set $512\times512px$ image patches are therefore extracted from random image locations and persisted on disk in order to speed up the training. The probability of a pixel in this image patch representing the center of an non covered cell is modeled by a normalized probability density function (pdf) of a bivariate normal distribution, with a scalar covariance matrix of 128, with its center at the nearest non covered cell. The value of this pdf at the position of the pixel of the image patch is its probability representing the center of an non covered cell. As image augmentation is used in the training process, see part 3.4 (*Training*), it is not yet determined which is the center pixel of an image patch. Therefore an input image patch is not paired with a target value but with a complete label image. The label image has the same dimensions as the input image and has a pdf just described, with its center at every center of a non covered cell in the corresponding input image. For every pixel, that is not covered by a pdf value, has value 1e−05 assigned. Which is, in the following, referred to as 'background value'. To map the target value into $(0, 1]$, the label is divided by it's maximum value. The actual target value – the center pixel of the label image – is extracted at run time after the eventual augmentation of the input image and label image.
The number of extracted image patches per comb image is 5000. This results in a total of 320.000 $512\times512px$ image patches, occupying approximately 160GB of disk space. Which is feasible and sufficient. It was experimented with different covariances for the pdf – in the following referred to as 'cov. pdf.', as well as with different background values. Two kinds of data sets were generated. Some with uniform sampling, which leads to more image patches of background. And some with weighted sampling, from now on referred to as 'weighted data set'. Which means that image coordinates, which are the centers of the patches to be extracted, are assigned a probability of being sampled, which in turn correlates with the corresponding value of their label image. This leads to more image patches having a non covered cell in their center. The results of these sampling techniques are depict in figure 21, which shows the respective distributions of target values.

3.4 Training

The data set is initially divided into a training set and a test set. Therefore the 64 annotated images are split, roughly 85/15 train/test, into a set of 54 and a set of 10 images, from which the 5000 image patches each are sampled from for the respective set. This ensures that the network does not see any of the test images in the training process.
As loss function PyTorch's Binary Cross Entropy with Logits is used. This is more or less equivalent to the use of sigmoid activation on the final layer and the subsequently application of Binary Cross Entropy loss, but with higher numerical stability. Which means the output of the final conv-layer is squeezed to a value between 0 and 1, repre-

senting the probability of a pixel being center of a non covered cell.
The network was trained using Stochastic Gradient Descent (SGD) with Nesterov Momentum[41] with a base learning rate of 0.0125, scheduled by Cosine Annealing[28] without warm restarts. Training was done for a fixed number of batches, with batch size 32. Since training was performed on the personal machine of the author (Intel Core i5-3470, Nvidia GTX 1070, 32GB RAM), computing time was very precious. Hence, the possibilities for experimenting with batch size were limited. Therefore, it was decided from the beginning to stick with a batch size of 32. The number of batches was either 100.000 or 200.000. Which corresponds to almost 12 and more than 23 epochs respectively.
Early stopping is performed by saving the network weights every time a new minimum of the running average, with a window size of 1000, of the validation loss is discovered. For this purpose, for each training batch a batch of the test set is evaluated. Which isn't the clean way, since no separate validation set is used, but would eventually lead to the best model before overfitting kicks in. No hyperparameters are optimized based on validation with the test set.

To achieve the desired robustness of the localizer model, as mentioned in part *Motivation* of section *Introduction*, some augmentation techniques were applied during training, using the Python library imgaug[12]. The augmentations were used carefully and with caution. They should not look too extreme – in a way still natural – but should simulate a wide range of appearance that may occur in reality. Therefore a range of intensities was defined for each of the methods using stochastic parameters. The augmentation of one training sample then consisted of a randomly selected combination of the individual methods in a randomly selected intensity each. The following augmentations, together with their probability of application:

- horizontal flip, prob. 0.5
- vertical flip, prob. 0.5
- affine transformation (incl. scaling, shearing, rotation), prob. 0.75
- gaussian blur, prob. 0.2
- sharpening, prob. 0.5
- linear contrast (inc./dec.: 0.5/0.5), prob. 0.5
- additive gaussian noise, prob. 0.5

, were applied exactly in this order and are depict in figure 8, which reflects the boundaries of their intensity ranges. In the following these augmentations are referred to as 'advanced augmentation'. In contrast, 'simple augmentation' refers to horizontal and vertical flips only.
All affine transformations were applied with bicubic interpolation and a constant background value of zero. One augmented batch of the training data set for the localizer is depict in figure 9. An important detail to consider was the implementation of augmentation with respect to the label image. As the label for a training sample gets extracted from the label image, as explained in part *Ground Truth Data*, on the fly, after the augmentation process, exactly the same augmentations applied to the input image has

[12] https://github.com/aleju/imgaug [Online, accessed January 15, 2022]

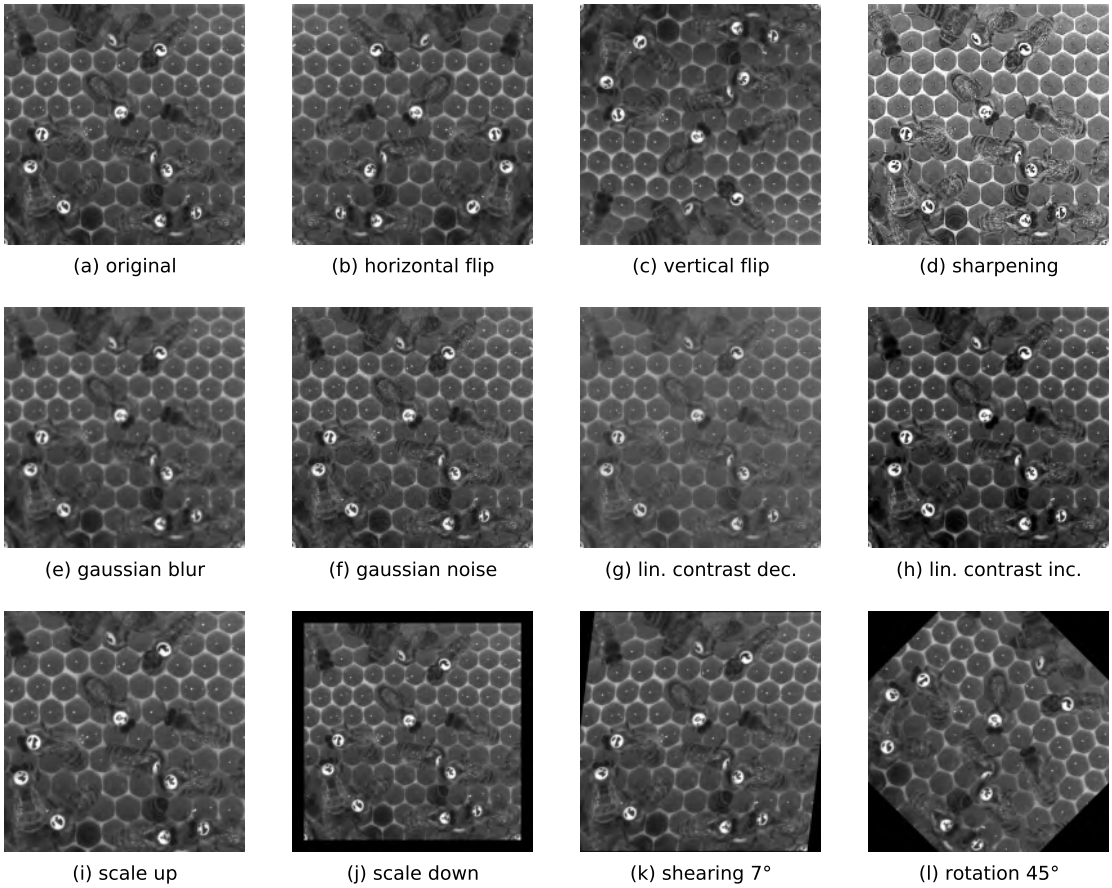
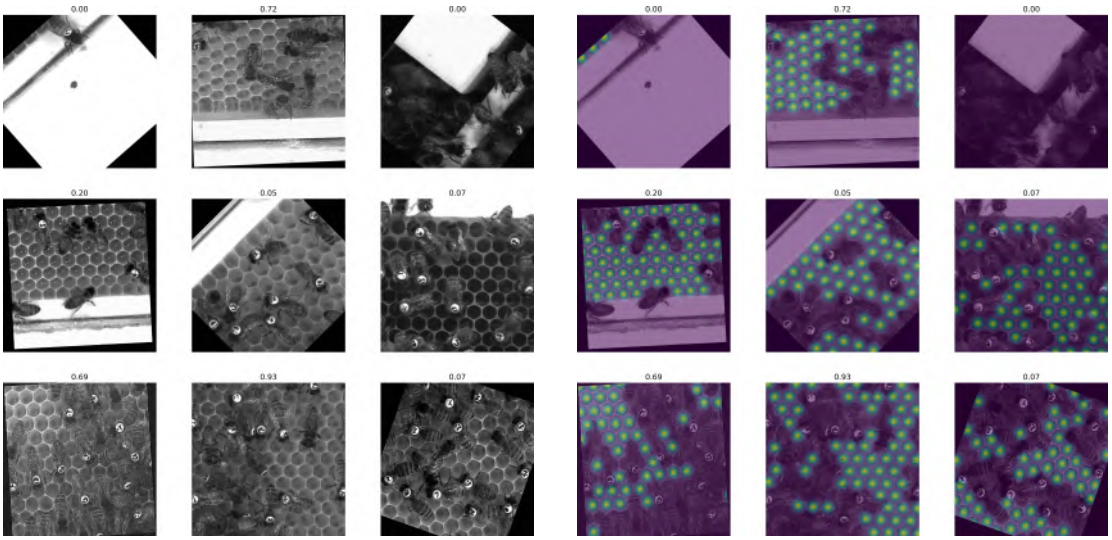


Figure 8: Augmentation Methods Used in the Training Process of the Localizer Model. **(a)** shows an original training sample. The other 11 images show augmented versions of **(a)**. The augmentations were applied within a strictly defined range of intensities. A random combination of several augmentations was applied to each of the training samples. The augmentations are shown here at their highest intensity each as used for training. In particular, these are: **(b)** a horizontal or left-right flip, **(c)** a vertical or upside down flip, **(d)** sharpening, here with an *alpha* value of 1.0 (full intensity), **(e)** gaussian blur, here with a std. deviation of 1.5 for the gaussian kernel, **(f)** gaussian noise, here with a mean of 0 and std. deviation of 12.75, **(g)** a linear reduction of contrast, here with multiplier 0.75, **(h)** a linear increase of contrast, here with multiplier 1.25, **(i)** upscaling, which is effectively zooming in, here with scaling factor 1.1, **(j)** downscaling, which is effectively zooming out, here with scaling factor 0.9, **(k)** shearing, here +7° degrees on the x-axis, **(l)** rotation around image center, here with +45° degrees.

to be applied to the label image. Which only counts for the affine transformations and image flips. Since these can cause the content in the center of the image to change. All methods that only change the value of individual pixels, but leave the shape and semantics of the overall image unchanged – namely blurring, sharpening, contrast changes and the introduction of noise – were not allowed to be transferred to the label image, since they could thus directly alter the label to be extracted.

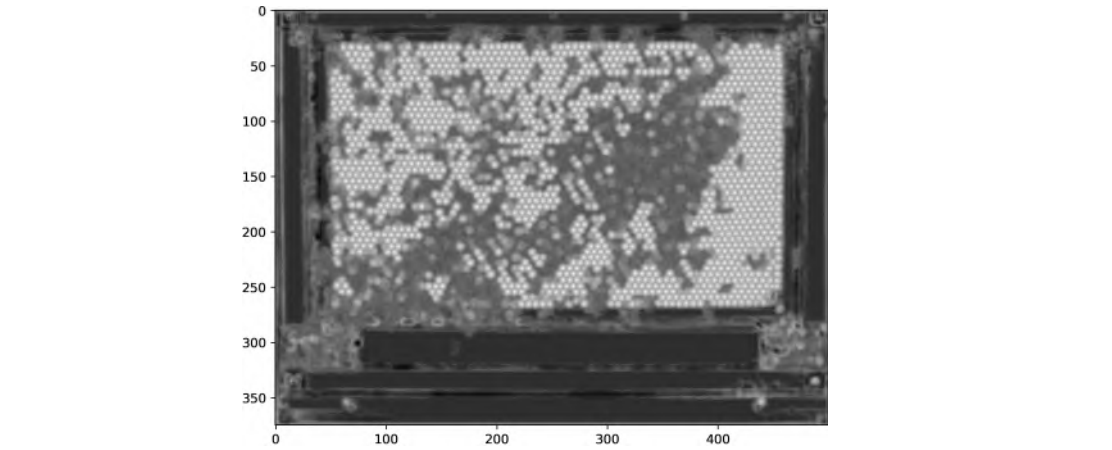


(a) (b)

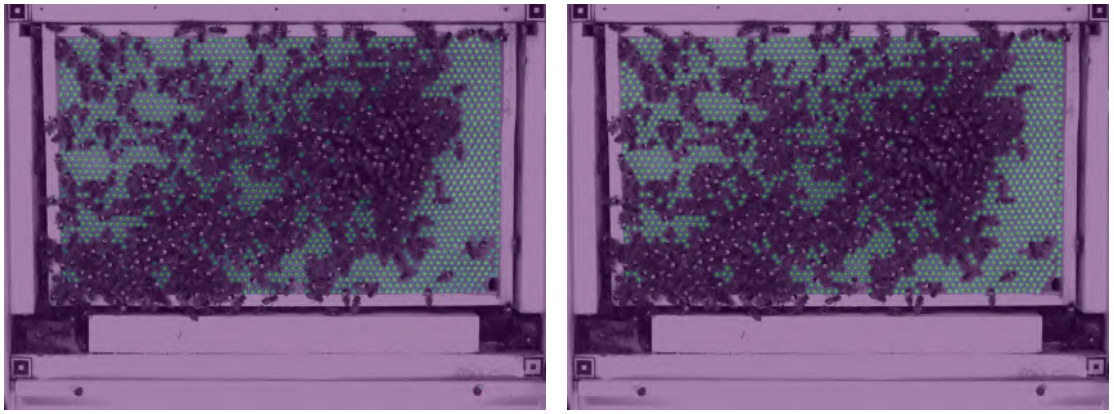
Figure 9: Augmented Training Samples for Localizer Model. **(a)** A batch of nine augmented input images, which represent the combination of several single augmentation methods depict in figure 8. **(b)** The same batch as in (a), but overlaid with the label image. The label image has the same dimensions as the input image and has a normalized probability density function of a bivariate normal distribution, with a scalar covariance matrix of 128, with its center on every center of a non covered cell on the input image. The actual label for an input image, in terms of the target for the error function, is extracted at runtime in the training process after the augmentation of both the input and label image and simply corresponds to the pixel value at the label image center.

3.5 Inference and Post-Processing

In order to obtain image coordinates of the non covered cells of a comb image, the network has to be applied in inference mode on the input image. First, the input image is suitably padded to compensate for the pixel losses at the image edges by convolution without padding. And additionally, the sigmoid function is applied to the output of the network to squeeze all values into the interval [0,1] – the probabilities. This produces an output like shown in figure 10 (a). The output can be seen as a saliency map, from which the positions of the centers of the predicted pdfs, if you will, get extracted via local maximum filtering. With calculated centroids of larger image sections of the local maxima, the accuracy of their positions is increased to sub pixel level, before the coordinates are upscaled to the native resolution of the input image.



(a)



(b) (c)

Figure 10: Localizer Inference. **(a)** Output of Localizer model with dimension 375×500. Brighter pixels mean higher values, i. e. probabilities. In contrast to training mode, in inference mode the sigmoid function has to be applied to the output in order to squeeze all values into [0,1] **(b)** input image with overlaid inference image (a), resized to input size 3000×4000 **(c)** input image with overlaid label image (ground truth).

4 Larval Age Classifier

For the deep neural network architecture of the brood classifying model, in the following just referred to as 'Classifier', two different approaches in terms of task modeling are tested. The first approach models the prediction of brood age as a regression task – regression on the brood age on an hourly scale. The second approach models it as a classification task – classification of the brood age into the discrete 217 classes representing the age in hours from 0 hours – an egg which is just been laid – to 216 hours – a larva just before its cell is capped. Thereby both models have the ability to say that there is no brood in the input image at all. Which is crucial, because the localizer simply extracts cells in which a brood would most likely be seen if it were present. With no guarantee that there is actually bee brood.

4.1 Data Set

For the creation of the training data for this network, preliminary work from [14] can be used. The data set that can be accessed includes 52 background images with annotated capped cells – brood cells of larvae in the capped larval stage. These background images are generated as described in [48].

The first step of creating the ground truth data is to determine which cell is being capped at which point in time. With this information we can back-calculate how old brood in a specific cell at a specific point in time must be. Hence first, background images of consecutive days have to be identified. That are images, that are at most 24 hours, rounded to full hours, apart, because the time stamp of capping of the cell should be determined with a certain precision. The time stamp is always provided in the file name of an image. From the set of 52 images 35 are consecutive. From these 35 images, 26 image pairs of consecutive images, the first one is the earlier one and the second the later one, can be created. Next, KD-Trees are used to determine which cell is capped in the second image but not yet in the first one. This procedure now leads to 916 detections of newly capped cells. Figure 24 depicts the first 5 detections of 4 image pairs. Depending on how close the respective image pairs are to each other, the calculated time of capping is as accurate. Thereby the maximum time span is 24 hours. The minimum is 3 hours, as the background images are calculated with images over a time span of 3 hours. As time of capping, in the end, simply the time stamp of the second image is used. Since the combs are replaced regularly, it was necessary to determine beforehand exactly when these times were. The explained procedure was applied only within two such exchanges. This is the only way to ensure that the respective cells keep the same coordinate.

The second step of creating the ground truth data encompasses the sampling of images from the recording season 2019 of the observation hive (see figure 4 for example image), the application of the Localizer onto these samples and the subsequent calculation of the age of the brood in the respective detected non covered cells, in case they are capped at some point in the future, based on the data from the first step. Always within the time limits for the replacement of the honeycomb. About 104.000 images are sampled evenly distributed. The procedure just described is now applied to each of these images, where the age of the brood in a specific cell is calculated using Eq.2. This value is negative for empty cells, in which the egg is not yet laid. These samples are needed, because the model will learn to not only classify the age of brood, but to also to tell brood from no brood. Other cells cannot be chosen as empty cell samples, because only for cells, that will be capped within a certain period of time, we know for sure, that they do not contain brood for more than 9 days prior to capping.

$$\text{Brood Age} = 9\text{days} - \text{timespan}(\text{timestamp\_cell\_capped} - \text{timestamp\_of\_image}) \quad (2)$$

During this procedure triplets of image name, coordinate and age in hours are saved in order to separate the ground truth data from the image data. Figure 25 depicts the spatial distribution of all samples, that are processed with this method.

Only in the third and last step it is iterated over the resulting triplets from step 2, saving the corresponding image patches, together with the age as target value, in H5 file format. The saved image patches are of size $192 \times 192 px$. Negative age values are clipped to -1, so that every 'no brood' sample is labeled with -1. This provides enough

head room for image augmentations such as rotation or cropping, with no image areas left blank in the final 128×128px network inputs. Because image patches of empty cells

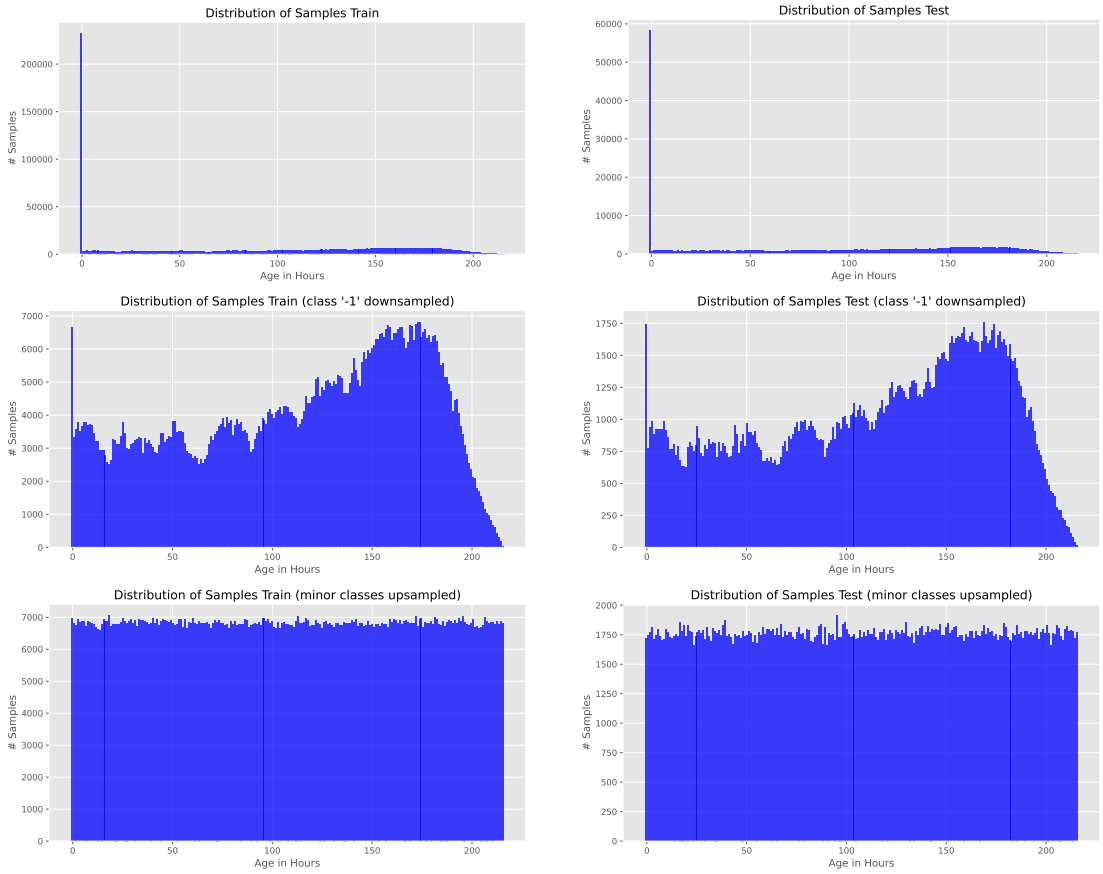


Figure 11: Distribution of Labels in Classifier Data Set. The 2 steps for the creation of a balanced data set are shown for **(a)**, the training set and for **(b)**, the test set. All bar plots show histograms of brood age classes, with the age in hours on the x-axis and the corresponding number of samples on the y-axis. The native data set (upper figures) is dominated by class '-1', no brood. The fewest samples are for the oldest larvae, right end of x-axis. First class '-1' is randomly subsampled to the number of samples of the most represented of all the remaining classes (result: middle). Thereafter, all underrepresented classes are aligned with these two via random oversampling (result: lower figures). This results in a well balanced data set. However, with the drawback of the most underrepresented classes being oversampled very heavily, consisting only of a few original samples.

are always saved too, the data set, in the following referred to as 'native data set', is very imbalanced with respect to the classes representing cells with brood, in the following referred to as positive classes, versus the class '-1', i. e. 'empty cell'. 27.3% of all samples are of class '-1'. In order to introduce no unwanted bias to the model and get better classification results the data set is rebalanced as depict in figure 11, such that every class is represented more or less in the same proportion.
A caveat to this data set is that the most underrepresented classes, the classes '200

hours' and older are being oversampled very heavily. Hence all samples of these classes consist of only of a few original samples duplicated multiple times. It remains to be seen whether this has a negative learning effect or accuracy drop for these classes.
Another caveat is, that theoretically, in the resulting data set, the label 0 corresponds to the exact time of laying the egg, on the premise that the time until a cell gets capped is exactly 9 days. Since in practice this is not the case, plus only non covered cells were sampled – the queen opaques a cell, when laying an egg – an image patch with this label will more likely show an egg than it will show an empty cell. This is due to the initial ground truth data provided by the work of [14], which was not designed to solve the problem of this work. The capped cells were annotated at a maximum frequency of 3 hours. In this case a specific annotated cell was capped sometime within those 3 hours. And in this best case, a time span of 3 hours between two annotated images, the image patch will likely show an egg, which is 1.5 hours old, when assuming a normal distribution with a mean of 9 days for the period of time passing between the laying of the egg and the capping of the corresponding cell. In short, we can notice the capping only too late, never too early. Therefore an image patch with label 0 will most likely show an egg, a few hours old, inside of the cell. But sometimes even an empty cell, due to variations in the length of the egg or open larval stage. This blurring of individual classes can make it difficult for the network to learn well.
In addition to that, the quality of the samples is not always good either. From occasional manual reviews of the samples, there are even more difficult examples than easy ones. The author is of the opinion that it is almost impossible to recognize an egg on the images. He has not been able to see one so far. But he did not look for it specifically. In figure 27 arbitrarily selected samples are shown. From each superclass, 'no brood', 'egg', 'larva', one easy and one difficult example. And in the case of the egg, it is impossible to see an egg even with a good view of the cell interior. The question is whether the egg can't stand out in color, is hidden by the sidewall of the cell, or if there is simply no egg in it yet. Theoretically, on fig.27(b) upper, there is a 7 hour old egg in the middle cell.

4.2 Discarded Regression Model

The first approach – the regression model – is quickly discarded by the author, because he does not achieve to get the network to learn anything meaningful. The architecture itself is similar to the one described in 4.3, with the difference of reducing the number of channels of the final conv-layer output to 1. With which the network outputs a single value – the predicted age in hours. With the option to output -1 in the case, that there is no brood in the presented cell. As loss mean squared error (MSE) is used. However, all models, trained with different hyperparameters, see figure 23 tend to simply predict the mean of the training data distribution. Which appears to be a common problem with regression models. At least for those trained with imperfect data sets. 'Classifier' always refers to the model described in 4.3.

4.3 Multiclass Classification Model

The basic model of the Classifier is derived from the Localizer model, see part 3.2 of section *Non Covered Cell Localizer*, hence uses a fully convolutional ResNet architecture with fixup initialization. Which enables the application of this network to arbitrarily sized inputs, as long as the input has a certain minimum size. The network consists of

8 2-D conv-layers, see figure 12, with a kernel size of 3×3 each. 6 of them are embedded in 3 'fixup-layers' of 1 residual block each. A residual block consists of 2 conv-layers, of which the output of the second layer is concatenated with the input for the first layer[52, 17]. The 3 fixup-layers are prepended with an initial conv-layer, which reduces image dimensions right at the start, and are followed by a final conv-layer, which inflates the number of channels to the number of classes, 218 – the classes −1 to 216. So the network outputs a one-dimensional vector whose entries are interpreted as probabilities for the corresponding classes. The predicted class is then obtained by the index of the maximum value in this vector. Downsampling is performed by the initial conv-layer and also by the first conv-layers of every fixup-layer by applying convolution with a stride of 2. No padding is applied, which is why image dimensions get reduced by conv-layers even without explicit downsampling. The whole network encompasses 542.266 learnable parameters.

4.4 Training

The data set is initially divided into a training set and a test set with ratio 80/20. Since the native data set gets balanced via random super- and subsampling of classes, the separation is done before this balancing procedure in order to prevent any data leakage from one to the other set, as depict in figure 11. Resulting in 1.496.875 training and 374.219 test samples – a total of 1.871.094 samples. The training procedure was the same as for the Localizer, see part 3.4, except for a few details. As loss function PyTorch's Cross Entropy Loss is used, which implicitly applies a softmax activation of the network output. The base learning rate for SGD is 0.1 for batch sizes of 256 or 512. For batch size 32 the base learning rate has to be reduced to 0.0125. The learning rate again is scheduled by Cosine Annealing without warm restarts, as this rather prevents the network from converging well, due to less training time with very low learning rate – assumption made by the author. Training is done for a fixed number of batches. For the 12 final training runs with different hyperparameters the networks are trained for either 25, 50 or 75 epochs. This is possible thanks to the kind permission to use Curta[3] for the training runs as well as for training data storage and sharing with the team of the Biorobotic Lab.
Image augmentation is also performed similar to 3.4. Additionally to the methods depict in figure 8 it is experimented with contrast limited adaptive histogram equalization (CLAHE) and random crops. Random crops were only applied in a restrained manner, ensuring the cell of interest is still roughly in the middle of the image patch. The maximum deviation in pixels to the center of the image, 'maximum crop shift', was tested with 3 to 10 pixels. An augmented batch of training samples are depict in figure 28.
In contrast to the regression model, via multiclass classification typically the the semantics of the individual classes, the order of the age value in this case, get lost. Each class is seen as completely distinct from the others. In order to give the network a kind of sense for the natural order of the classes label smoothing is introduced to the training process, as depict in figure 13, using a gaussian filter with standard deviation of 3 for the filtering of the one-hot encoding of the class. During training the target classes are encoded in the range of [0, 217]. On inference the classification value is then obtained by the index of the maximum value in this vector minus one.

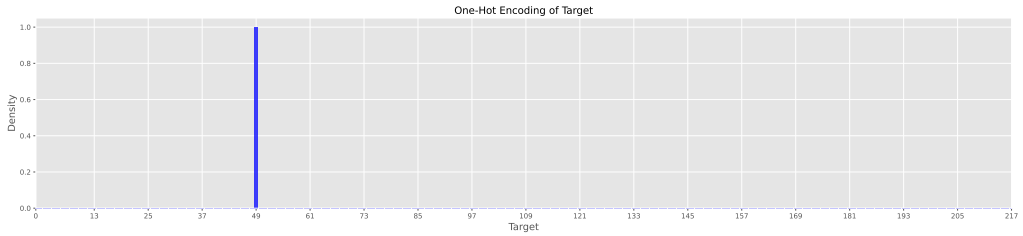
```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [256, 16, 63, 63]             160
              ReLU-2          [256, 16, 63, 63]               0
            Conv2d-3          [256, 32, 31, 31]           4,640
              ReLU-4          [256, 32, 31, 31]               0
            Conv2d-5          [256, 32, 29, 29]           9,248
              ReLU-6          [256, 32, 29, 29]               0
    FixupBasicBlock-7          [256, 32, 29, 29]               0
            Conv2d-8          [256, 64, 14, 14]          18,496
              ReLU-9          [256, 64, 14, 14]               0
           Conv2d-10          [256, 64, 12, 12]          36,928
             ReLU-11          [256, 64, 12, 12]               0
   FixupBasicBlock-12          [256, 64, 12, 12]               0
           Conv2d-13          [256, 128, 5, 5]           73,856
             ReLU-14          [256, 128, 5, 5]               0
           Conv2d-15          [256, 128, 3, 3]         147,584
             ReLU-16          [256, 128, 3, 3]               0
   FixupBasicBlock-17          [256, 128, 3, 3]               0
           Conv2d-18          [256, 218, 1, 1]         251,354
================================================================
Total params: 542,266
Trainable params: 542,266
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 16.00
Forward/backward pass size (MB): 648.55
Params size (MB): 2.07
Estimated Total Size (MB): 666.62
----------------------------------------------------------------
```

Figure 12: Classifier Model. Output of the function `summary` from the library torchsummary[a]. Shape corresponds to [batch size, channels, height, width]. The network consists of 3 'fixup' layers. Each 'fixup' layer consists of one residual block (indicated by 'FixupBasicBlock-X', stated after each residual block in summary), which in turn consists of 2 conv-layers. Input dimensions for the model, in this case, are [256, 1, 128, 128]. Downsampling in image dimensions is performed by 'Conv2d-1' and by the first conv-layers of every 'fixup' layer, 'Conv2d-3', 'Conv2d-08' and 'Conv2d-13' by operating with a stride of 2. Otherwise stride is 1. The last layer 'Conv2d-18' inflates the number of channels to the number of classes, 218 – classes −1 to 216. This outputs a one-dimensional vector whose entries are interpreted as probabilities for the corresponding classes. The classification value is then obtained by the index of the maximum value in this vector. During training, all class labels are increased by 1, in order to make the label compatible with indexing. Which leads to labels in range $[0, 217]$, see figure 13. On inference, the final classification value is obtained by a subsequent reduction, of the classification value, by 1.

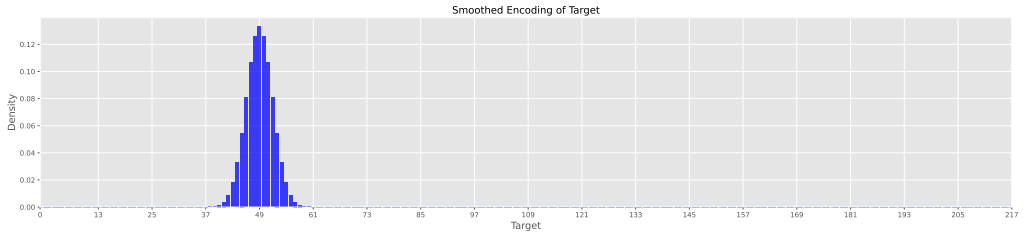
[a] https://pypi.org/project/torchsummary/ [Online, accessed January 24, 2022]

5 Results

This section provides the final results of both the Localizer and the Classifier.



(a)



(b)

Figure 13: Label Smoothing for Classifier Model. Both plots depict an encoding of the target vector, representing class '49', for the training of the Classifier model via Cross Entropy Loss. **(a)** shows the One-Hot-Encoding of the target. **(b)** shows the smoothed encoding, which distributes the density to 24 neighboring classes or hours respectively. The smoothing is obtained by filtering the One-Hot-Encoding (a) with a gaussian kernel with standard deviation of 3. This is to give the network a sense of the order of the classes.

5.1 Localizer

The model of the Localizer was trained with 12 different combinations of hyperparameters and data set properties. In all cases the model quickly converged to a point, where the loss almost reached it's final mark, improving only marginal afterwards. See the training history of the best model in figure 22.

All trained models were evaluated on the basis of the metrics precision (Eq.3), recall (Eq.4), F_1 score (Eq.5), F_2 score (Eq.6) and 'mean distance' – the mean distance of the predicted from the ground truth location in pixels. The evaluation data is summarized in table 3. Which considers each model operated with two different thresholds, one optimized for best F_1 and the second optimized for best F_2 score, as 2 different models. The definitions for the former metrics are:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3}$$

$$\text{Recall} = \frac{\text{TP}}{\text{P}} \tag{4}$$

$$\text{F}_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5}$$

$$\text{F}_2 = 5 \cdot \frac{\text{Precision} \cdot \text{Recall}}{4 \cdot \text{Precision} + \text{Recall}} \tag{6}$$

, where TP is the number of true positives (detected cells, that are indeed non covered cells), FP the number of false positives (number of detections for positions, where there is no cell or a covered or a capped cell) and P the number of annotated cells (condition positive).

The F_1 score is the harmonic mean of precision and recall, whereas the F_2 score weights the recall twice as much as precision. Hence F_2 is what should be looked at, when high sensitivity is more important than high precision.

The best model was selected based on the best F1 score. Since this model is likely to be continuously applied to the hive, increasing the likelihood that each non covered cell will be detected every once in a while. Even if the model is not overly sensitive. In this case it is the model (row 3 of table 3), that was trained on a weighted data set with a cov. pdf. of 128 and with advanced augmentation, as depict in figure 9. The model was trained the full number of 200.000 batches, which corresponds to more than 23 epochs. Early stopping, implemented as described in 3.4, did not have a positive impact on the final evaluation metrics. The weights for the early stopped model were saved only about 2000 batches before the model was fully trained. The metrics for the early stopped model differs from the fully trained model only in subtle amounts. Which was to expect due to the curve of the training and validation loss in the training history depict in figure 22. Both were constantly descending. Which makes early stopping unnecessary. At a threshold of 0.5643 this model yields a F_1 score of 0.9599 at a precision of 0.9644, which is the best among all models. Mean distance is $\sim$ 2.4px. At a threshold of 0.2674 this model yields the best F_2 score of all models. With a mean distance of $\sim$ 2.4px. See figure 14 for for it's precision-recall curve, figure 15 for it's F_1 curve and figure 16 for it's F_2 curve. However, the best recall is achieved by a model, see table 3 row 12 and 20, that was trained on an unweighted dataset that uses a cov. pdf. of 50 for only 100.000 batches. With a mean distance of only $\sim$ 1.5px.

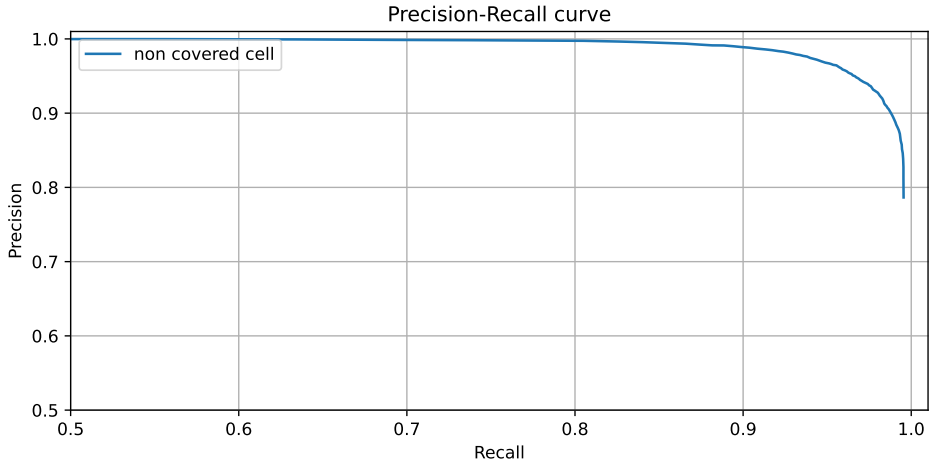


Figure 14: Precision-Recall Curve of Localizer Model. The blue line shows the tradeoff between precision and recall, one has to give up one for the other, for different thresholds. A high area under the curve (AUC) represents both high recall and high precision. The perfect model would have both a recall and precision of 1.

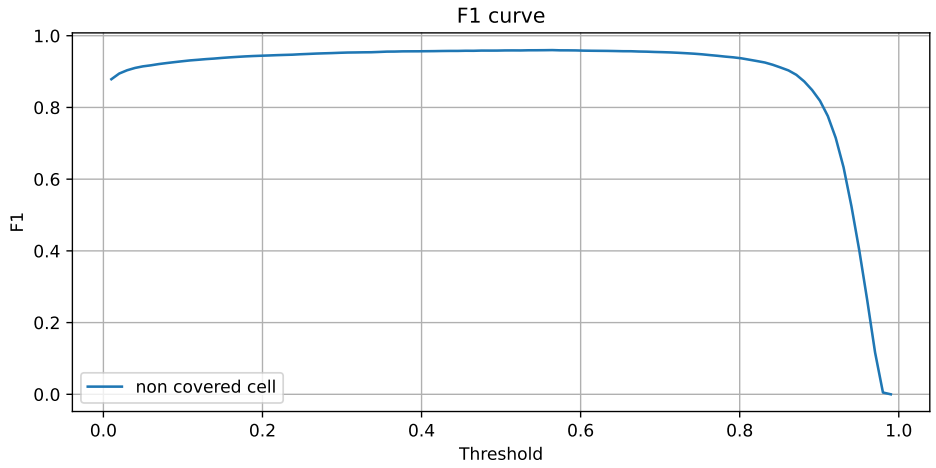


Figure 15: F_1 Curve of Localizer Model. The blue line shows the F_1 score, y-axis, depending on the used threshold, x-axis, for the model. F_1 score has it's maximum at threshold 0.5643. This is the threshold being used at the inflection point of the precision-recall curve in fig.14.

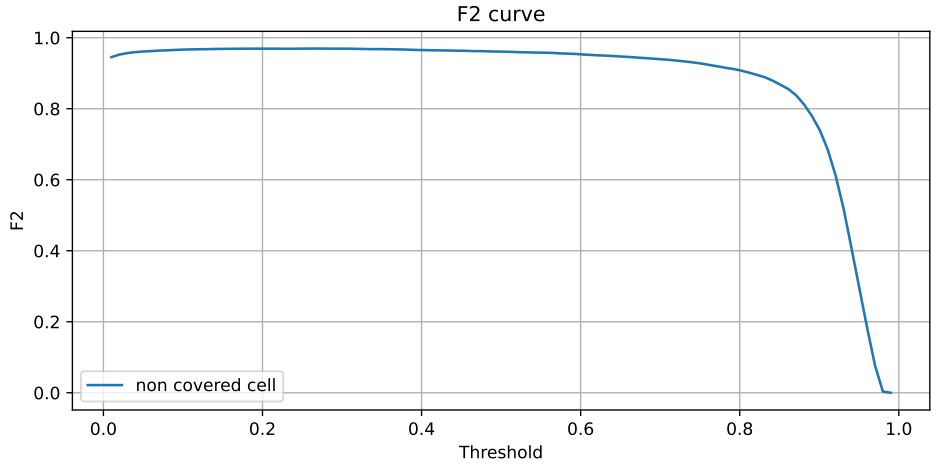
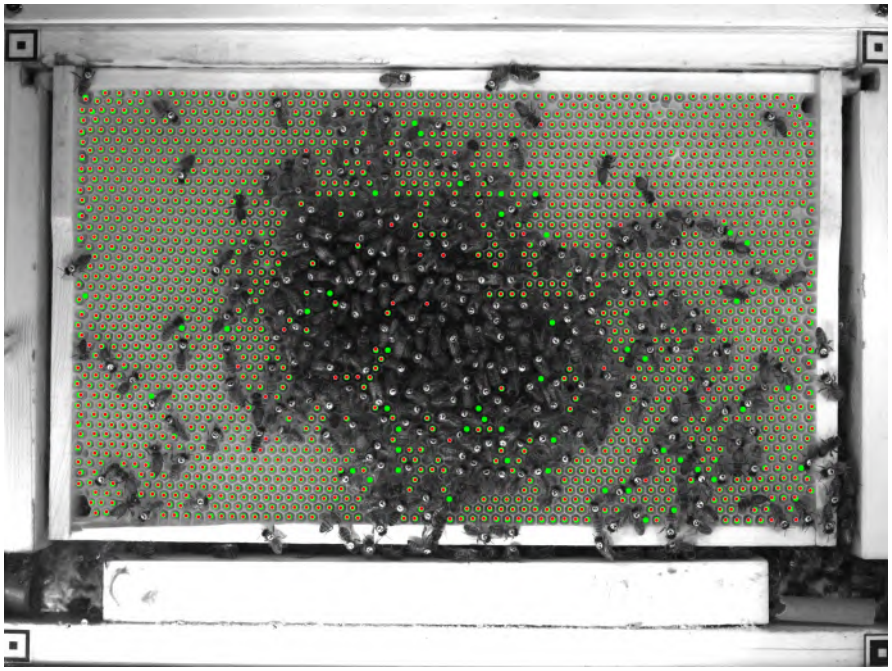


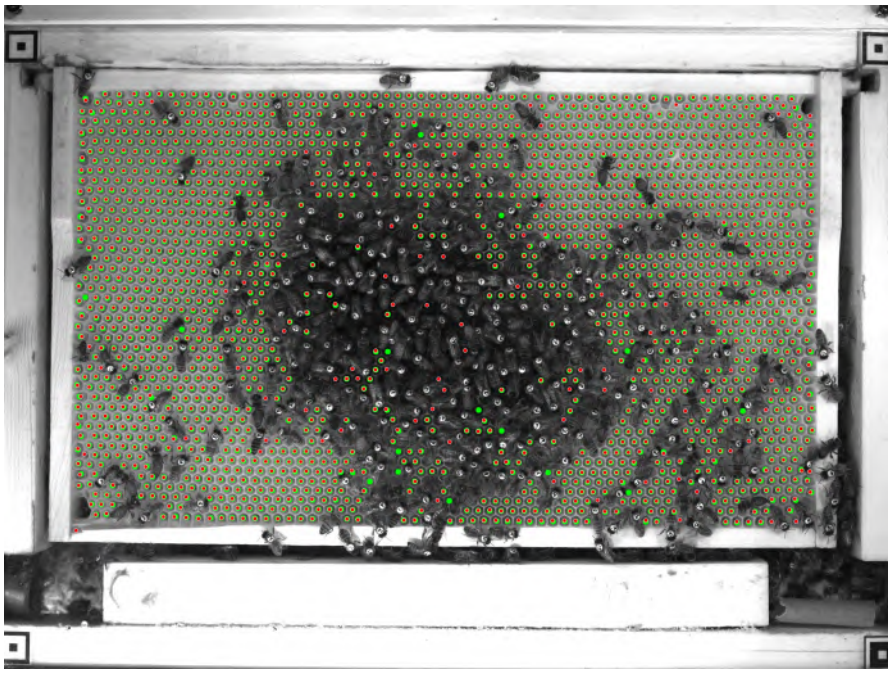
Figure 16: F_2 Curve of Localizer Model. The blue line shows the F_2 score, y-axis, depending on the used threshold, x-axis, for the model. F_2 score has it's maximum at threshold 0.2674.

See figure 17 for a visualization of the output of the best Localizer model, comparing the F1-optimized (using threshold 0.5643) and the F2-optimized (using threshold 0.2674) operation mode. Note here, that in both cases, with one exception in fig.17(b) lower left corner, false positives are still valid detections of open cells. Hence the evaluation metrics better should be considered in conjunction with the corresponding ground truth data. They correlate with the quality of the ground truth data.

The whole processing of one single input image, including pre-processing (padding,



(a)



(b)

Figure 17: Visualization of Localizer Output with Different Thresholds. Showing ground truth locations in green color and Localizer output in red. **(a)** F1-optimized operation mode, using threshold 0.5643, which yields a precision of 0.9644 and a recall of 0.9555. **(b)** F2-optimized operation mode, using threshold 0.2674, which yields a precision of 0.9198 and a recall of 0.9827.

substraction of mean, division by standard deviation, loading to GPU memory) and

post-processing (applying sigmoid, transfer to RAM, maximum filtering, calculation of maxima patch centroids, mapping to original coordinate space) takes about $680ms$ on average. Whereby the pure inference time, running the network on the preprocessed input, is only $2ms$. All measurements are done on the author's personal machine (Intel Core i5-3470, Nvidia GTX 1070, 32GB RAM).

5.2 Classifier

The model is evaluated two-fold. On the one hand, the model is evaluated completely on the balanced test set according to common regression metrics. On the other hand, the model is evaluated purely as a binary classifier, for the classes brood and no brood on the native test data set, which comprises 21% 'No Brood' samples, since this is probably more in line with the actual distribution of inputs when combined with the Localizer. Which outputs a big proportion of empty cells. The errors made by the classifier on this data set are plotted on a heatmap and contrasted with the automatically determined annotations.

By the time this work is completed, not all networks are done with training. Therefore, an extensive evaluation with a comparison of all 12 trained models must be omitted here. Instead, only the most promising model, i.e. the one with the lowest training loss, which has been completed so far, is considered here. And is referred to as the best model.

The best model is trained with label smoothing, advanced augmentation with a maximum crop shift of 3px, but without CLAHE, a batch size of 256 for 75 epochs. See figure 29 for the training history. By the end of the training the training loss is 4.0182 and validation loss is 3.9884 (Cross Entropy). Validation loss is always lower as training loss for training with advanced augmentation.

Although trained as multiclass classifier, the error the model produces is evaluated by applying the metrics mean absolute error (MAE), mean squared error (MSE), root mean squared error (RMSE) as well as mean and standard deviation.

Balanced Test Set The results for the evaluation of the model as is on the balanced test set are are summarized in table 1. A visualization of the model's performance is provided in figure 18

Table 1: Evaluation Metrics of the Classifier on the Balanced Test Set.

Metric	Value
mean	2.06
std.	26.53
MAE	15.88
MSE	708.25
RMSE	26.61

Native Test Set The results for the evaluation of the Classifier as a binary classifier, classifying into 'brood' and 'no brood', are summarized in table 2. The metrics comprise

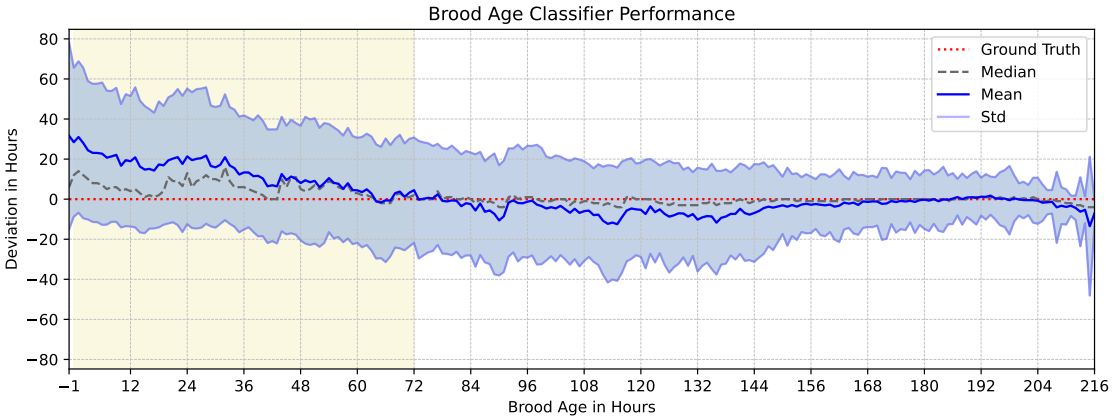


Figure 18: Classification Error of the Classifier Model on the Balanced Test Set. The deviation from the ground truth values for the respective classes, with egg stage highlighted yellow. The highest errors, in terms of absolute regression errors, of the model are produced for the egg stage. Here the age of the egg is predicted way to high. For the larval stage area, 72 to 216 hours, a negative correlation of the error for a certain class with the number of training samples for the same class in the native data set, see figure 11 middle, can be detected.

recall, precision, F_1 score, F_2 score and – to emphasize the shortcomings of this classifier as a binary classifier, which are else shaded by the pure proportion of positive samples, i. e. imbalanced data set, making the usual metrics misleading – the false positive rate based on the count of true positives (TP), false postives (FP), true negatives (TN), false negatives (FN). A visualization of the model's performance is provided in figure 19 in form of a confusion matrix.

Table 2: Evaluation Metrics for the Classifier as a Binary Classifier on the Native Test Set. All evaluation metrics applied to the Classifier as a binary classifier, classifying into 'brood' and 'no brood'

Metric	Value
TP	218555
FP	48829
TN	9339
FN	13
precision	0.82
recall	0.99
false positive rate	0.84
F_1	0.90
F_2	0.96

All errors the Classifier made on the native test set are depict in figure 26 in the same manner as done for the ground truth data in figure 25. It can be spotted, that the highest errors done for the class 'egg' are made in peripheral areas, indicating that the view on the eggs is obstructed by the side walls of the cells. The distribution of errors for the other classes are my due to the pure distribution of test samples in the native test set.

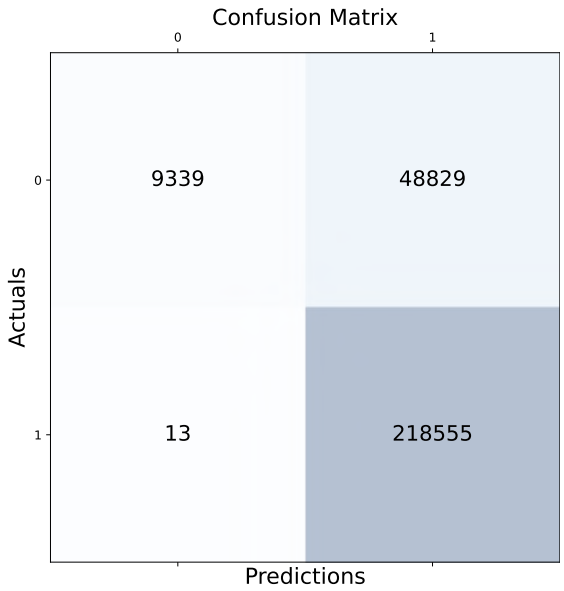


Figure 19: Confusion Matrix for Classification Model as a Binary Classifier on the Native Test Set. Confusion Matrix visualizing the performance as binary classifier into 'brood' (1) and 'no brood' (0). Note, that the number of false positives is much higher than the number of true negatives.

6 Conclusion

This section draws a conclusion to the evaluation of the final results, assesses the potential benefits in practice and gives an outlook on how the results can be improved through future work.

6.1 Discussion of Results

With a F_1 score of 0.9599 and F_2 score of 0.9573 at a precision of 0.9644 and a recall of 0.9555, the Localizer is well balanced and capable of performing the task set for it. But it would have to work faster in order to be applicable in real time, which is 6Hz at the moment. Hence the time for complete processing of one image has to be reduced to about $160ms$. And it would have to be reduced even further when running in parallel with other modules on one machine.

The classifier does a solid job on predicting the brood age of well visible brood, i. e. older larvae. It does not so well on images of younger larvae and eggs. The latter mainly due to the poor visibility. With a prediction error of 2.1 ± 26.5 hours (mean $\pm$ standard deviation) it is of by 0.66 days on average. The bad results for the Classifier as a binary classifier, predicting 'brood cell' or 'empty cell' is most likely due to the confusion of class 'egg' with class 'empty cell'. Since in most cases the image shows the same for both classes, because the egg is too small to see it or is hidden by the side walls of the cells.

6.2 Future Work

The processing time of the Localizer can only be greatly improved by optimization of the pre- and post-processing steps. It may help to forego the increase of accuracy via calculation of image moments. If enough GPU memory were available, it would also be beneficial to run the entire module in graphics memory to reduce inter memory reads and writes. Doing post-processing in C instead of Python is also an option.
Precision and recall of the Localizer could be improved by improved ground truth data. The network is likely to face conflicting examples during training. Revised ground truth data should be more consistent in the way a cell is annotated or not. Uniform rules for whether a cell should be labeled, for example, if an antenna or leg is in front of it.
Completely different architectures should also be tried out. Or even different concepts. If the network could also detect different classes of cells, in terms of their content – brood, honey, nectar, pollen, an adult bee – would allow to sort out everything except brood beforehand. Which makes it easier for the Classifier plus post-processing is only done for brood cells, which in turn would improve processing time.

The Classifier could be improved with better training data, which is quality controlled by human experts. Due to variations in the length of the egg and larval stages labels of the data set can simply be false. The data set itself, meaning the images of the observation comb, should be improved and tailored to this task. Although the image data is highly resolving, the system would benefit from cameras whose focus is set on the interior of the cells and not on the circular tags of the bees. The lighting could also be improved, so that it covers the interior of the cells better. Since most samples suffer from too low contrast in the interior of the cell.
In summary, a classifier for honey bee brood age requires brighter and sharper images of the brood. Maybe with the help of a dedicated camera, with it's focal point on the spatial center of the cells and improved lighting. But there is head room for better training techniques too. Training with bigger image patches in order to provide more meta data to the classifier is worth a try. This could induce a helpful bias through the surrounding cells. Since the nest is usually used in a dense fashion.

References

[1] Wario et al. "Automatic detection and decoding of honey bee waggle dances". In: Public Library of Science, 2017. Submitted.

[2] Thiago S. Alves et al. "Automatic detection and classification of honey bee comb cells using deep learning". en. In: *Computers and Electronics in Agriculture* 170 (Mar. 2020), p. 105244. ISSN: 01681699. DOI: 10.1016/j.compag.2020.105244. URL: https://linkinghub.elsevier.com/retrieve/pii/S0168169919307690 (visited on 01/18/2022).

[3] Loris Bennett, Bernd Melchers, and Boris Proppe. "Curta: A General-purpose High-Performance Computer at ZEDAT, Freie Universität Berlin". In: (2020). In collab. with Universitätsbibliothek Der FU Berlin and Universitätsbibliothek Der FU Berlin. Artwork Size: 5 S. Publisher: Freie Universität Berlin, 5 S. DOI: 10.17169/REFUBIUM-26754. URL: https://refubium.fu-berlin.de/handle/fub188/26993 (visited on 01/01/2022).

[4] Andreas Berg. *Detecting Honey Bee Trophallaxis in Trajectory Data*. en. Bachelor Thesis. July 2018.

[5] Franziska Boenisch. *Feature Engineering and Probabilistic Tracking on Honey Bee Trajectories*. Bachelor Thesis. Feb. 2017.

[6] Franziska Boenisch et al. "Tracking all members of a honey bee colony over their lifetime". In: *arXiv:1802.03192 [cs]* (Mar. 2018). arXiv: 1802.03192. URL: http://arxiv.org/abs/1802.03192 (visited on 01/18/2022).

[7] Katarzyna Bozek et al. "Author Correction: Markerless tracking of an entire honey bee colony". en. In: *Nature Communications* 12.1 (Dec. 2021), p. 3121. ISSN: 2041-1723. DOI: 10.1038/s41467-021-23297-4. URL: http://www.nature.com/articles/s41467-021-23297-4 (visited on 01/18/2022).

[8] Annely Brandt et al. "Immunosuppression in Honeybee Queens by the Neonicotinoids Thiacloprid and Clothianidin". In: *Scientific Reports* 7.1 (Dec. 2017), p. 4673. ISSN: 2045-2322. DOI: 10.1038/s41598-017-04734-1. URL: http://www.nature.com/articles/s41598-017-04734-1 (visited on 01/04/2022).

[9] S. Burrows et al. *Exotic Bee ID*. USDA APHIS Identification Technology Program (ITP) and Utah State University. Fort Collins, CO. 2021. URL: http://idtools.org/id/bees/exotic/factsheet.php?name=16736 (visited on 01/01/2022).

[10] Daniel P. Cariveau et al. "The Allometry of Bee Proboscis Length and Its Uses in Ecology". en. In: *PLOS ONE* 11.3 (Mar. 2016). Ed. by Olav Rueppell, e0151482. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0151482. URL: https://dx.plos.org/10.1371/journal.pone.0151482 (visited on 12/28/2021).

[11] K. Crailsheim et al. "Collective and individual nursing investment in the queen and in young and old honeybee larvae during foraging and non-foraging periods". In: *Insectes Sociaux* 50.2 (Apr. 2003), pp. 174–184. ISSN: 0020-1812, 1420-9098. DOI: 10.1007/s00040-003-0644-x. URL: http://link.springer.com/10.1007/s00040-003-0644-x (visited on 01/18/2022).

[12] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[13] Richard O. Duda and Peter E. Hart. "Use of the Hough transformation to detect lines and curves in pictures". en. In: *Communications of the ACM* 15.1 (Jan.

1972), pp. 11–15. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/361237.361242. URL: https://dl.acm.org/doi/10.1145/361237.361242 (visited on 01/21/2022).

[14] Paulo Engelke. *Erkennen von gedeckelten Brutzellen in Bienenwaben*. Bachelor Thesis. Feb. 2021.

[15] Nicola Gallai et al. "Economic valuation of the vulnerability of world agriculture confronted to pollinator decline". In: *Ecological Economics* 68 (Jan. 2009), pp. 810–821. DOI: 10.1016/j.ecolecon.2008.06.014.

[16] Elke Genersch et al. "The German bee monitoring project: a long term study to understand periodically high winter losses of honey bee colonies". In: *Apidologie* 41.3 (May 2010), pp. 332–352. ISSN: 0044-8435, 1297-9678. DOI: 10.1051/apido/2010014. URL: http://link.springer.com/10.1051/apido/2010014 (visited on 01/01/2022).

[17] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *arXiv:1512.03385 [cs]* (Dec. 2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385 (visited on 01/13/2022).

[18] Sepp Hochreiter. "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions". en. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 06.02 (Apr. 1998), pp. 107–116. ISSN: 0218-4885, 1793-6411. DOI: 10.1142/S0218488598000094. URL: https://www.worldscientific.com/doi/abs/10.1142/S0218488598000094 (visited on 01/13/2022).

[19] Andrew G. Howard et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". In: *arXiv:1704.04861 [cs]* (Apr. 2017). arXiv: 1704.04861. URL: http://arxiv.org/abs/1704.04861 (visited on 01/21/2022).

[20] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG].

[21] Hubert Koll. *Stationenlernen "Die Honigbiene"*. Deutscher Imkerbund e.V., Villiper Hauptstr. 3, 53343 Wachtberg, www.deutscherimkerbund.de. 2013. eprint: https://deutscherimkerbund.de/userfiles/Kinder_Jugendseite/Bienen_Extras/Honigbiene_Stationen_lernen_Web.pdf. (Visited on 01/08/2022).

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[23] Tim Landgraf et al. "Dancing Honey bee Robot Elicits Dance-Following and Recruits Foragers". In: *arXiv:1803.07126 [cs]* (Mar. 19, 2018). arXiv: 1803.07126. URL: http://arxiv.org/abs/1803.07126 (visited on 11/26/2020).

[24] Gretchen LeBuhn and Joshua Vargas Luna. "Pollinator decline: what do we know about the drivers of solitary bee declines?" In: *Current opinion in insect science* (2021).

[25] Y. LeCun et al. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Comput.* 1.4 (Dec. 1989), pp. 541–551. ISSN: 0899-7667. DOI: 10.1162/neco.1989.1.4.541. URL: https://doi.org/10.1162/neco.1989.1.4.541.

[26] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[27] Yann LeCun and Yoshua Bengio. "Convolutional Networks for Images, Speech, and Time Series". In: *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA, USA: MIT Press, 1998, pp. 255–258. ISBN: 0262511029.

[28] Ilya Loshchilov and Frank Hutter. *SGDR: Stochastic Gradient Descent with Warm Restarts*. 2017. arXiv: 1608.03983 [cs.LG].

[29] Wenfu Mao, Mary A. Schuler, and May R. Berenbaum. "A dietary phytochemical alters caste-associated gene expression in honey bees". In: *Science Advances* 1.7 (2015), e1500795. DOI: 10.1126/sciadv.1500795. eprint: https://www.science.org/doi/pdf/10.1126/sciadv.1500795. URL: https://www.science.org/doi/abs/10.1126/sciadv.1500795.

[30] Vinod Nair and Geoffrey E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Haifa, Israel: Omnipress, 2010, pp. 807–814. ISBN: 9781605589077.

[31] Simon Potts et al. "Global pollinator declines: Trends, impacts and drivers". In: *Trends in ecology evolution* 25 (Feb. 2010), pp. 345–53. DOI: 10.1016/j.tree.2010.01.007. eprint: https://facultyweb.cortland.edu/broyles/consem/Final%20Papers/global-pol-declines.pdf.

[32] Willem Proesmans et al. "Pathways for Novel Epidemiology: Plant–Pollinator–Pathogen Networks and Global Change". In: *Trends in Ecology & Evolution* 36.7 (July 2021), pp. 623–636. ISSN: 01695347. DOI: 10.1016/j.tree.2021.03.006. URL: https://linkinghub.elsevier.com/retrieve/pii/S0169534721000732 (visited on 01/04/2022).

[33] Raúl Rojas and Jerome Feldman. *Neural networks: a systematic introduction*. eng. Berlin Heidelberg: Springer, 1996. ISBN: 978-3-642-61068-4 978-3-540-60505-8. DOI: 10.1007/978-3-642-61068-4.

[34] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *arXiv:1505.04597 [cs]* (May 2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597 (visited on 01/21/2022).

[35] Christoph Sandrock et al. "Impact of Chronic Neonicotinoid Exposure on Honeybee Colony Performance and Queen Supersedure". In: *PLOS ONE* 9.8 (Aug. 2014), pp. 1–13. DOI: 10.1371/journal.pone.0103592. URL: https://doi.org/10.1371/journal.pone.0103592.

[36] a recognized scientist. "catchy title". In: *a fancy journal* ().

[37] Leon Sixt. *RenderGAN: Generating realistic labeled data – with an application on decoding bee tags*. Bachelor Thesis. Aug. 2016.

[38] M. L. Smith, M. M. Ostwald, and T. D. Seeley. "Honey bee sociometry: tracking honey bee colonies and their nest contents from colony founding until death". en. In: *Insectes Sociaux* 63.4 (Nov. 2016), pp. 553–563. ISSN: 0020-1812, 1420-9098. DOI: 10.1007/s00040-016-0499-6. URL: http://link.springer.com/10.1007/s00040-016-0499-6 (visited on 01/18/2022).

[39] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[40] Martina Stang et al. "Size-specific interaction patterns and size matching in a plant-pollinator interaction web". In: *Annals of botany* 103 (Mar. 2009), pp. 1459–69. DOI: 10.1093/aob/mcp027.

[41] Ilya Sutskever et al. “On the Importance of Initialization and Momentum in Deep Learning”. In: *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*. ICML’13. Atlanta, GA, USA: JMLR.org, 2013, III–1139–III–1147.

[42] Schmickl T. and Crailsheim K. “Cannibalism and early capping: strategy of honeybee colonies in times of experimental pollen shortages”. In: *Journal of Comparative Physiology A: Sensory, Neural, and Behavioral Physiology* 187.7 (Sept. 2001), pp. 541–547. ISSN: 0340-7594, 1432-1351. DOI: 10.1007/s003590100226. URL: http://link.springer.com/10.1007/s003590100226 (visited on 01/18/2022).

[43] J. Tautz et al. “Behavioral performance in adult honey bees is influenced by the temperature experienced during their pupal development”. In: *Proceedings of the National Academy of Sciences* 100.12 (June 10, 2003), pp. 7343–7347. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1232346100. URL: http://www.pnas.org/cgi/doi/10.1073/pnas.1232346100 (visited on 01/09/2022).

[44] Adam J Vanbergen and the Insect Pollinators Initiative. “Threats to an ecosystem service: pressures on pollinators”. In: *Frontiers in Ecology and the Environment* 11.5 (2013), pp. 251–259. DOI: https://doi.org/10.1890/120126. eprint: https://esajournals.onlinelibrary.wiley.com/doi/pdf/10.1890/120126. URL: https://esajournals.onlinelibrary.wiley.com/doi/abs/10.1890/120126.

[45] Fernando Wario et al. “Automatic detection and decoding of honey bee waggle dances”. In: *PLOS ONE* 12.12 (Dec. 2017). arXiv: 1708.06590, e0188626. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0188626. URL: http://arxiv.org/abs/1708.06590 (visited on 09/19/2020).

[46] Waugsberg. *Bienenwabe mit Eiern und Brut 5*. licensed under CC BY-SA 3.0, https://creativecommons.org/licenses/by-sa/3.0/legalcode. 2007. URL: https://en.wikipedia.org/wiki/File:Bienenwabe_mit_Eiern_und_Brut_5.jpg (visited on 01/16/2022).

[47] Benjamin Wild, Leon Sixt, and Tim Landgraf. “Automatic localization and decoding of honeybee markers using deep convolutional neural networks”. In: *arXiv:1802.04557 [cs]* (Feb. 2018). arXiv: 1802.04557. URL: http://arxiv.org/abs/1802.04557 (visited on 01/17/2022).

[48] Benjamin Wild et al. “Social networks predict the life and death of honey bees”. en. In: *Nature Communications* 12.1 (Feb. 2021). Number: 1 Publisher: Nature Publishing Group, p. 1110. ISSN: 2041-1723. DOI: 10.1038/s41467-021-21212-5. URL: https://www.nature.com/articles/s41467-021-21212-5 (visited on 02/17/2021).

[49] Geoffrey R. Williams et al. “Neonicotinoid pesticides severely affect honey bee queens”. In: *Scientific Reports* 5.1 (Dec. 2015), p. 14621. ISSN: 2045-2322. DOI: 10.1038/srep14621. URL: http://www.nature.com/articles/srep14621 (visited on 01/04/2022).

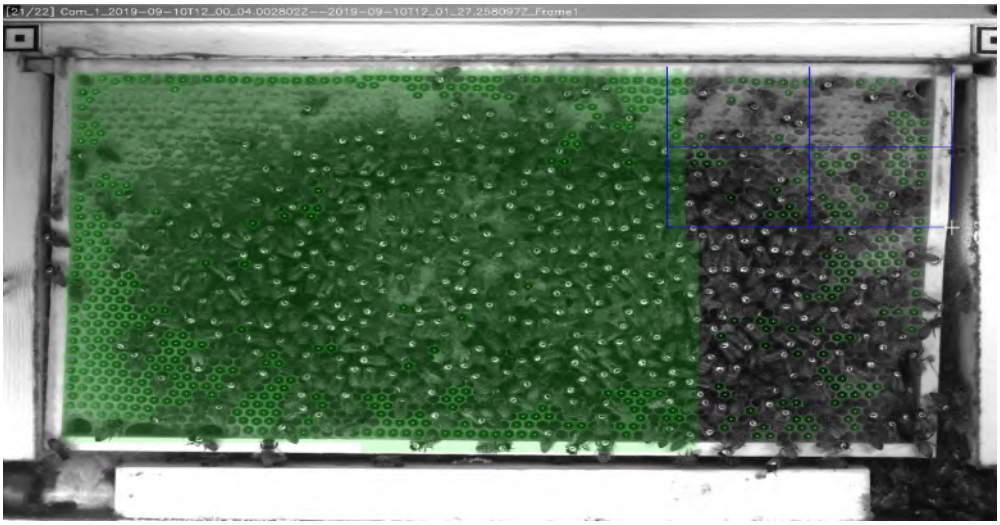
[50] M.L. Winston. *The Biology of the Honey Bee*. Harvard University Press, 1991. ISBN: 9780674074095. URL: https://books.google.de/books?id=-5iobWHLtAQC.

[51] Adrian Zachariae. “„Temporale Analyse der sozialen Netzwerke von Honigbienen“ ‘Temporal analysis of honeybee social networks’”. en. Master Thesis. 2017, p. 56.

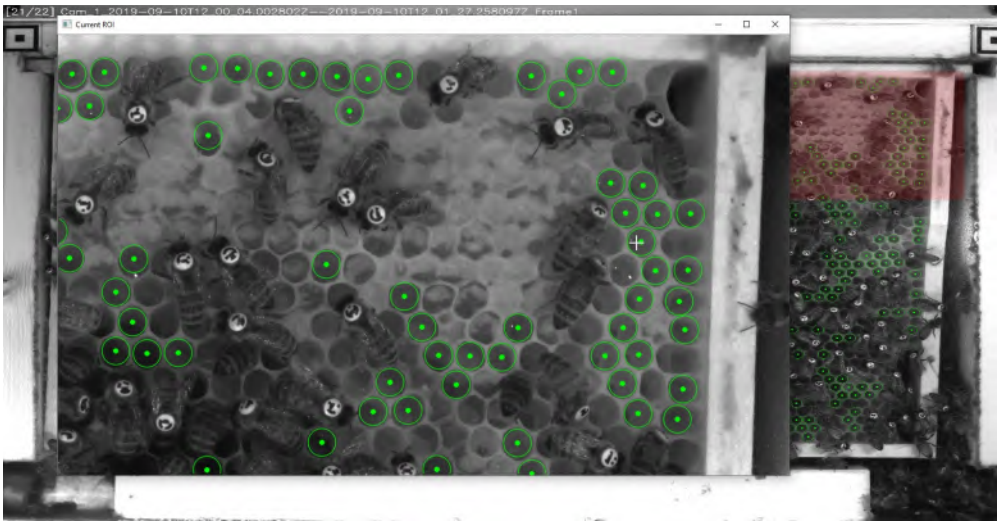
[52] Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. “Fixup Initialization: Residual Learning Without Normalization”. In: *arXiv:1901.09321 [cs, stat]* (Mar. 2019).

arXiv: 1901.09321. URL: http://arxiv.org/abs/1901.09321 (visited on 01/13/2022).

A Supplementary Figures



(a)



(b)

Figure 20: CenterClicker. **(a)** Screenshot of the tool CenterClicker, while selecting a ROI (blue rectangular in the upper right). Area which was selected as ROI before is highlighted in green. **(b)** Screenshot of the tool CenterClicker, with additional ROI window in the foreground, which renders the ROI in native resolution. Area which is currently selected as ROI is highlighted in red.

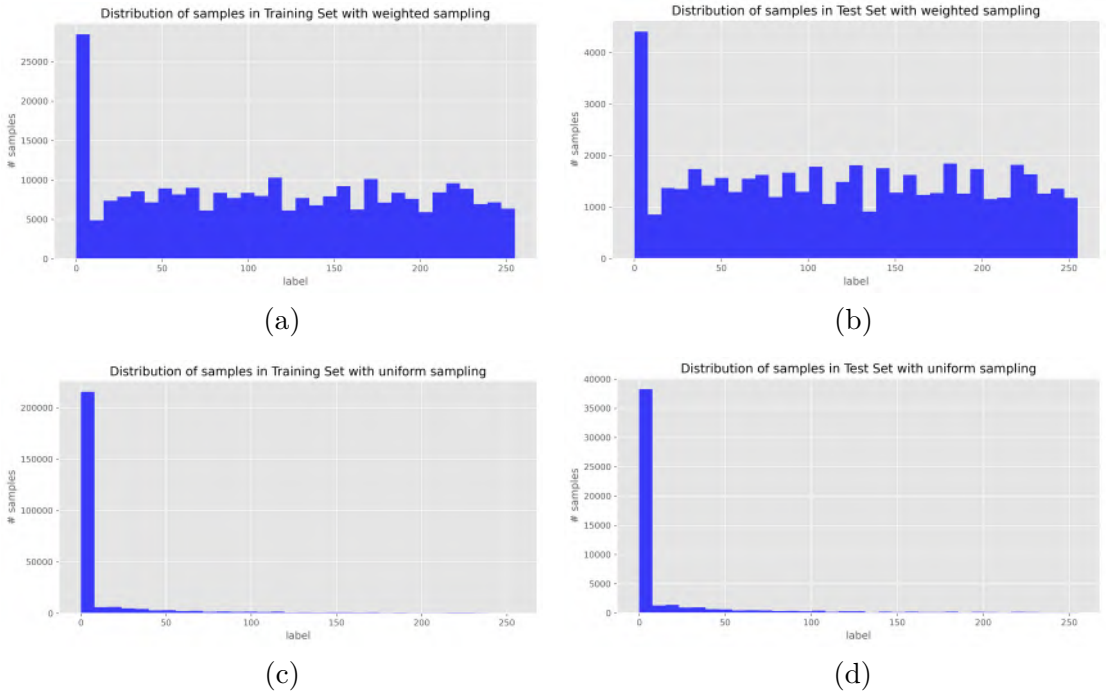


Figure 21: Distribution of Labels in Localizer Data Sets. All four bar plots show the histogram of the center pixels of all label images in the respective data set. The label on the x-axis corresponds to the pixel value in $[0, 255]$ of the 8-bit image. The actual label is then a normalized value in $(0, 1]$. The total number of occurrences is plotted on the y-axis. With weighted sampling, **(a)** and **(b)**, the data set is more balanced. Both data sets are generated with cov. pdf. of 128.

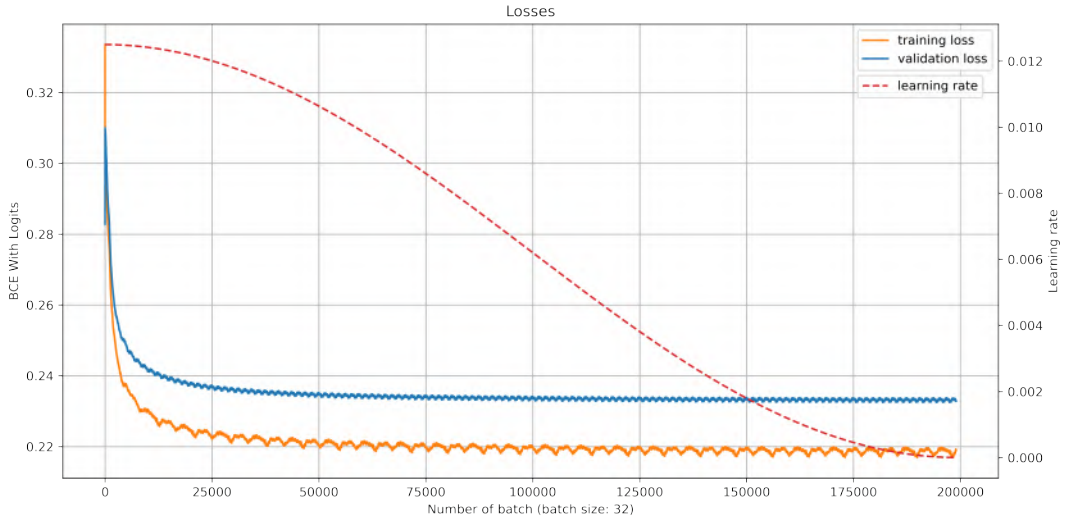
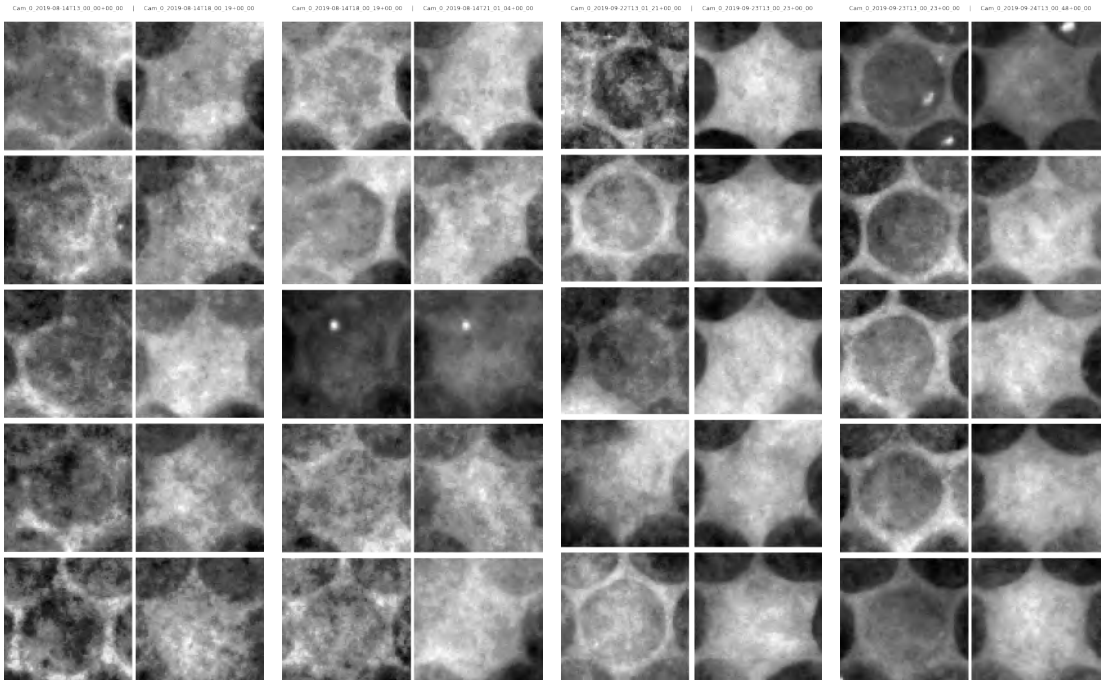


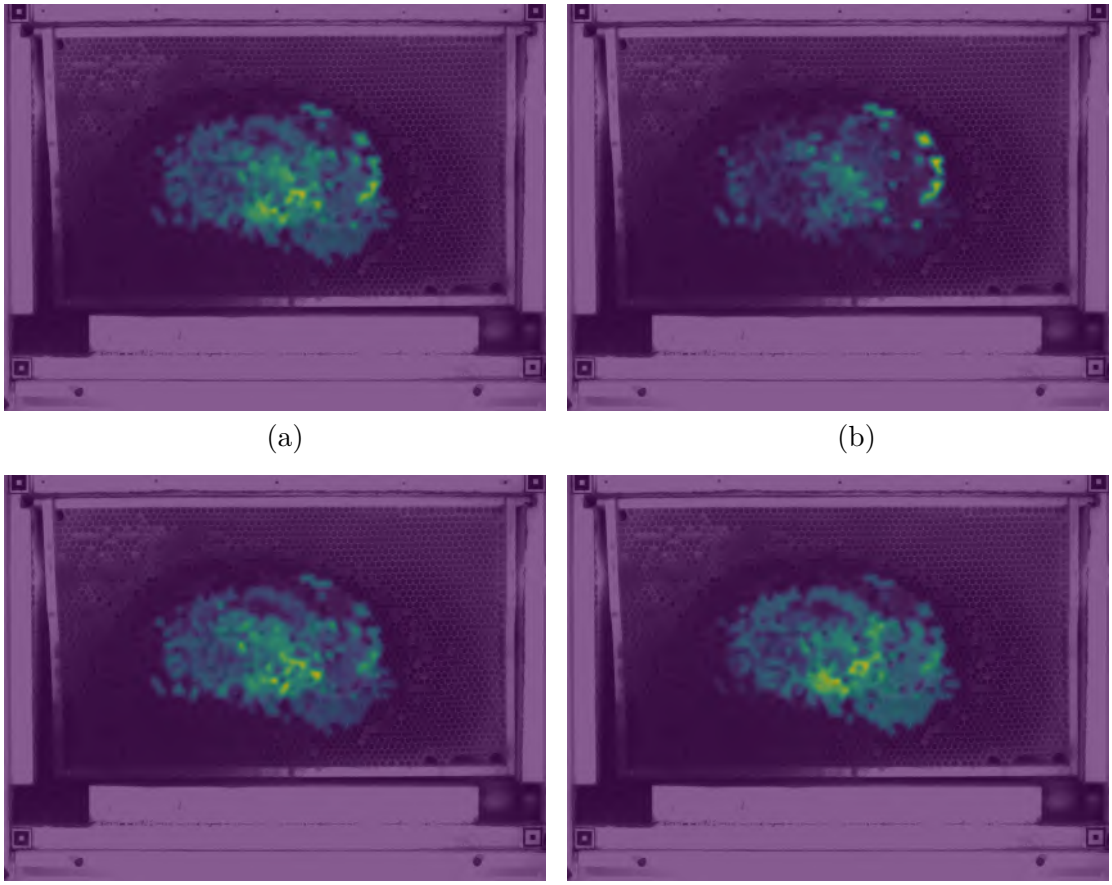
Figure 22: Training History of Localizer Model. Training and validation loss, smoothed with a running average with a window size of 1000 of the Localizer Model (Fixup-ResNet, fully convolutional) together with the learning rate, scheduled by Cosine Annealing. The model quickly, in about a quarter of the total training time, converged to almost the final validation loss, improving only marginal afterwards.

Figure 23: Regression Model Experiment with Hyperparameter Variations. y-axis: the minimum running average validation loss over the number of batches processed of 11 training runs with different combinations of hyperparameter values each. All with the same result – the early stagnation of the minimum running average validation loss, due to the model predicting always the mean of the training data.



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

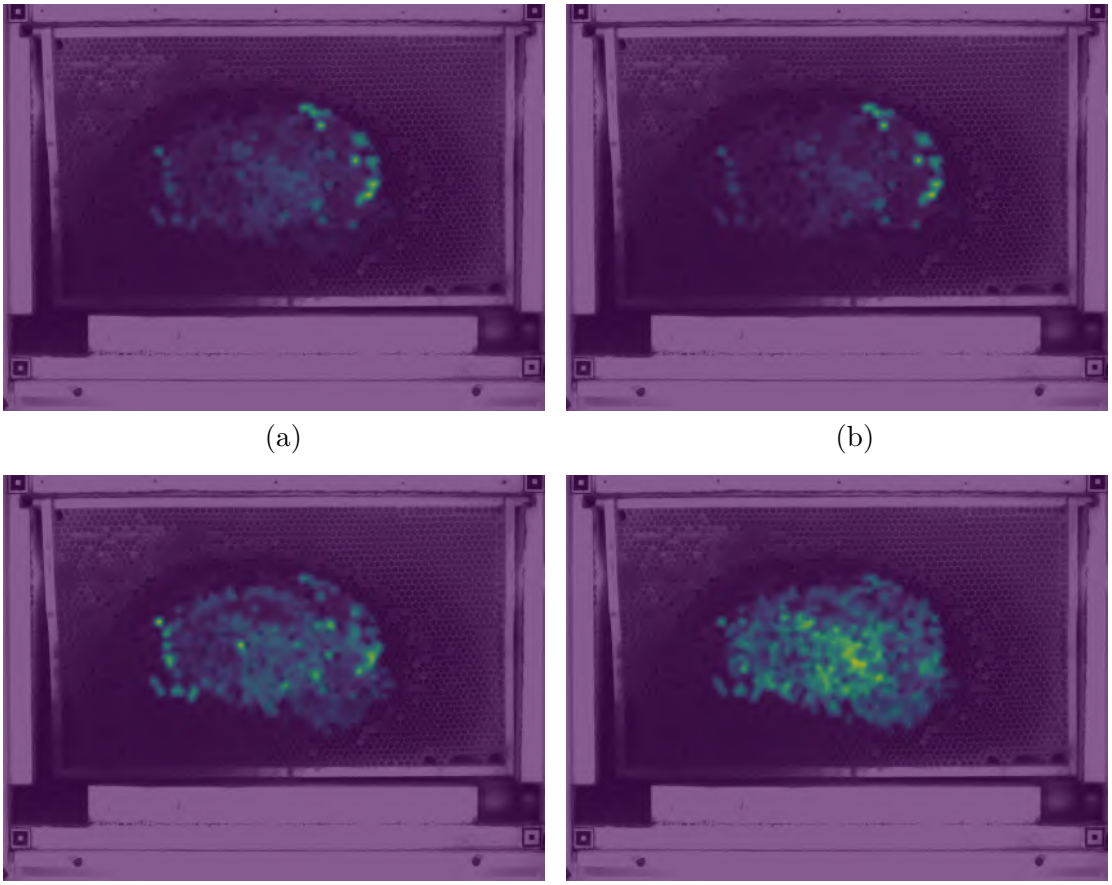
Figure 24: Newly Capped Cells. 5 uncapped cells each on the left side – same 5 cells, but now capped, on the right side. The capping of each of the cells happened sometime between **(a)** 2019-08-14 13:00:00 and 2019-08-14 18:00:19 **(b)** 2019-08-14 18:00:19 and 2019-08-14 21:01:04 **(c)** 2019-09-22 13:01:21 and 2019-09-23 13:00:23 **(d)** 2019-09-23 13:00:23 and 2019-09-24 13:00:48. The examples shown here, from Camera 0, are only the first 5 detections for each of the 4 time periods. Which are about 3 hours for **(a)** and **(b)** and about 24 hours for **(c)** and **(d)**.



(a) (b)

(c) (d)

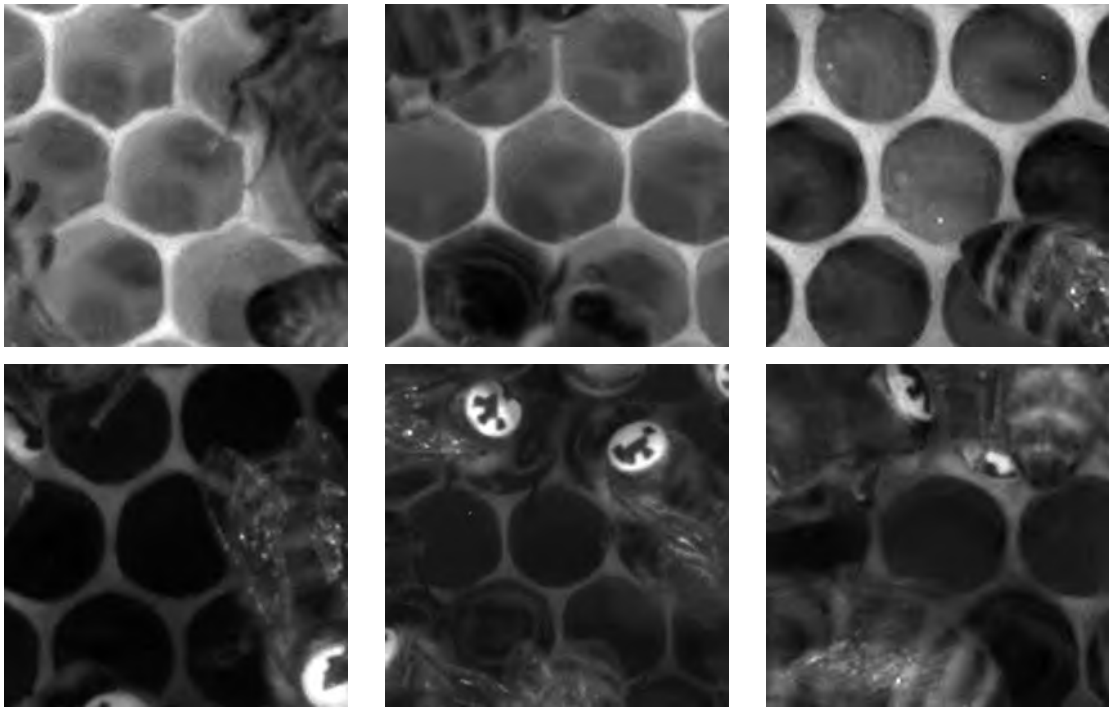
Figure 25: Spatial Distribution of Ground Truth Data for Brood Age. Proportions of automatically gathered ground truth data for brood age according to it's location on the comb, for **(a)** all classes together and separately, for the classes **(b)** 'no brood', **(c)** 'egg', **(d)** 'larva'. Brighter area means higher value, i. e. more samples from this location.



(a) (b)

(c) (d)

Figure 26: Spatial Distribution of MSE Errors Produced by the Classifier Model on the Native Test Set. Proportions of MSE errors according to it's location on the comb, for **(a)** all classes together and separately, for the classes **(b)** 'no brood', **(c)** 'egg', **(d)** 'larva'. Accumulations of the errors were normalized to [0,1]. Brighter area means higher value, i. e. more samples from this location.



(a) (b) (c)

Figure 27: Easy vs. Hard Data Set Samples for Classifier Model. **(a)** Empty cell; upper: 15h before egg laying, lower: 127h before egg laying **(b)** Egg; upper: 7h old, lower: 46h old **(c)** Larva; upper: 182h old, lower: 183h old. The upper pictures each show an easy example and the lower pictures a hard one. The problem with the latter is mainly the poor lighting. In addition to that, the focus of the cameras is set to the bee tags. The cell contents and especially the bottom of the cells are quite blurred. In both cases, one cannot see the egg.

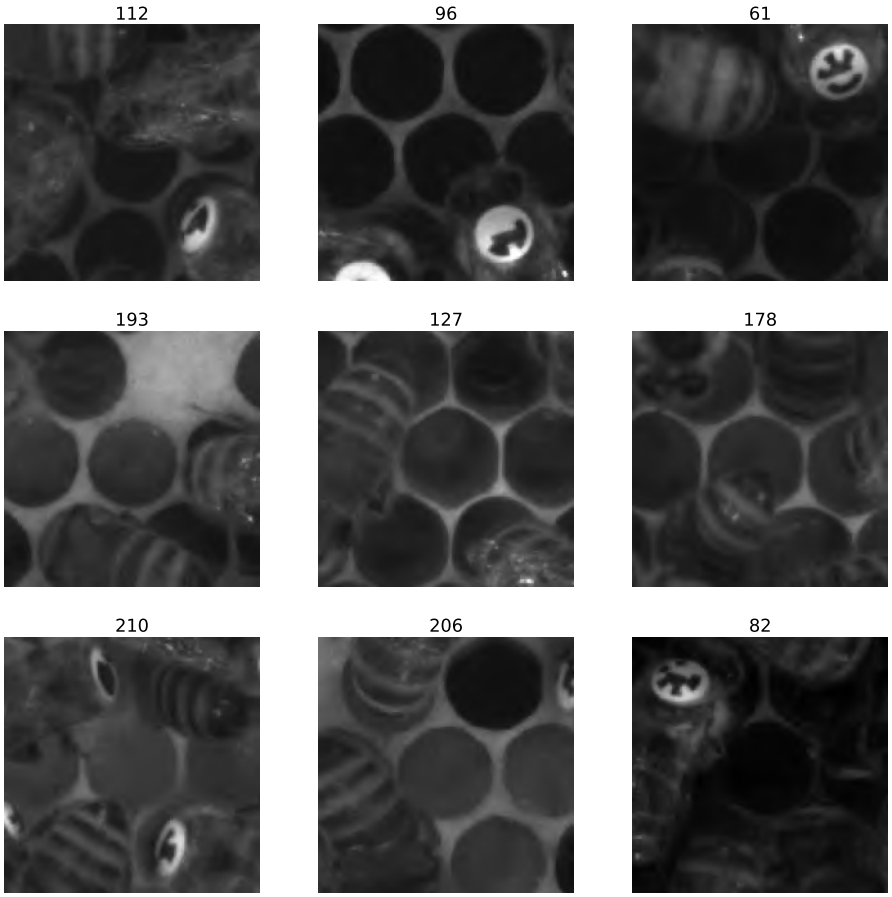


Figure 28: Augmented Training Samples for Classifier Model. A batch of nine augmented input images, titled with their label, of size 128×128px each. The label is the brood age in hours. Augmentation is applied in the same manner as for the training of the Localizer, see part 3.4 of section *Non Covered Cell Localizer*, encompassing the same types of augmentation, as shown in figure 8, but with slightly different combinations and probabilities of application. Additionally it was experimented with contrast limited adaptive histogram equalization (CLAHE) as augmentation technique. However, this has not proven to be advantageous.

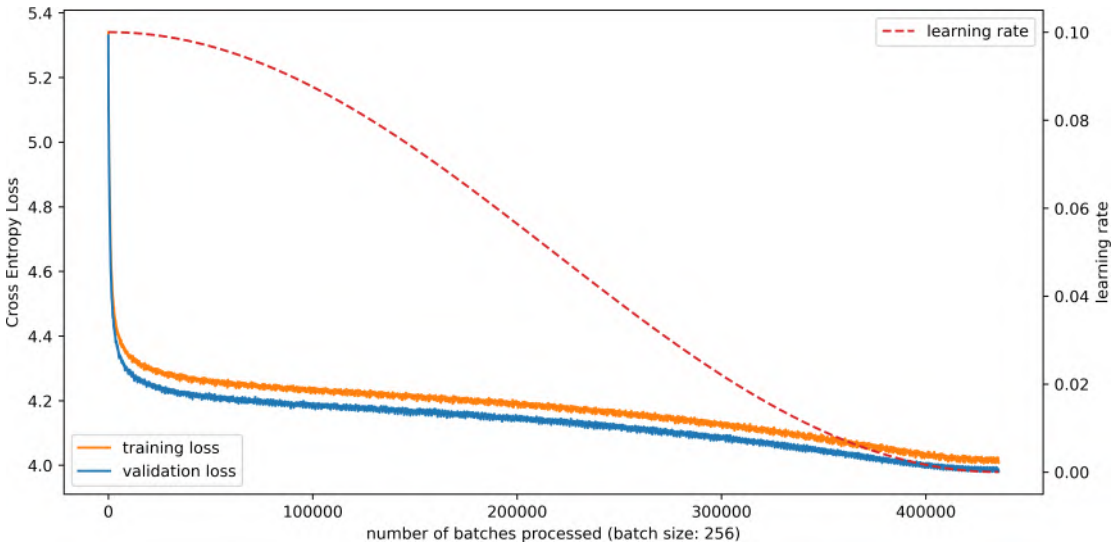


Figure 29: Training History of Classifier Model. Training and validation loss, smoothed with a running average with a window size of 250 of the Classifier Model (Fixup-ResNet, fully convolutional) together with the learning rate, scheduled by Cosine Annealing.

B Supplementary Tables

Table 3: Evaluation of 12 Trained Localizer Models. Common training parameters: batch size: 32, loss: BCE with Logits, optimizer: SGD, scheduler: Cosine Annealing. Common data set attributes: number training samples: 270.000, background value: 1e-05. The best result for each metric is shown in bold.

Data Set Attr.		Training Parameter			Evaluation Metrics					
cov. pdf	weighted sampling	adv. aug.	early stop.	nb. batches	threshold	precision	recall	F1	F2	mean dist.
128	✓	✓	✓	197917	0.5544	0.9630	0.9565	0.9597	0.9578	2.3581
128	✓	✓	✓	197917	0.2575	0.9178	0.9832	0.9494	0.9694	2.4004
128	✓	✓	✗	200000	0.5643	**0.9644**	0.9555	**0.9599**	0.9573	2.3572
128	✓	✓	✗	200000	0.2674	0.9198	0.9827	0.9502	**0.9695**	2.3997
128	✗	✗	✓	96740	0.5148	0.9594	0.9544	0.9569	0.9554	2.5128
128	✗	✗	✓	96740	0.1783	0.9040	0.9834	0.9421	0.9665	2.5812
128	✗	✗	✗	100000	0.5148	0.9604	0.9536	0.9570	0.9549	2.5053
128	✗	✗	✗	100000	0.1585	0.8992	0.9846	0.9400	0.9663	2.5799
50	✗	✓	✓	180000	0.3862	0.9440	0.9616	0.9527	0.9581	1.4182
50	✗	✓	✓	180000	0.1783	0.9009	0.9841	0.9407	0.9663	1.4537
50	✗	✓	✓	99745	0.4357	0.9449	0.9529	0.9489	0.9513	1.4874
50	✗	✓	✓	99745	0.1585	0.8817	**0.9851**	0.9305	0.9626	1.5461
50	✓	✓	✓	99086	0.4357	0.9471	0.9572	0.9521	0.9552	1.3911
50	✓	✓	✓	99086	0.2278	0.9027	0.9810	0.9402	0.9643	1.4256
50	✗	✓	✗	200000	0.3961	0.9434	0.9620	0.9526	0.9582	1.4180
50	✗	✓	✗	200000	0.1882	0.9003	0.9840	0.9403	0.9661	1.4531
50	✓	✓	✗	100000	0.4555	0.9512	0.9530	0.9521	0.9527	**1.3857**
50	✓	✓	✗	100000	0.2278	0.9027	0.9810	0.9402	0.9643	1.4257
50	✗	✓	✗	100000	0.4357	0.9449	0.9529	0.9489	0.9513	1.4877
50	✗	✓	✗	100000	0.1585	0.8817	**0.9851**	0.9305	0.9626	1.5461
50	✗	✗	✓	180000	0.4357	0.9553	0.9510	0.9531	0.9518	1.3942
50	✗	✗	✓	180000	0.1981	0.9106	0.9815	0.9447	0.9664	1.4400
50	✗	✗	✗	200000	0.4456	0.9556	0.9508	0.9532	0.9518	1.3952
50	✗	✗	✗	200000	0.1981	0.9089	0.9821	0.9440	0.9665	1.4421