IP Report: Assessment of honey bee colony health using CV and ML

Anvit Mangal

Winter 2019

1 Introduction

Honey bees are responsible for pollination and hence for most of the food that we consume. Honey bee colonies have been depreciating at an alarming rate and in 2018, 60% of the honey bees colonies in the United States have been vanished. This has tremendous impact on humankind and other organisms.

There are several factors as to why the bees are being affected so adversely and at such an alarming rate. Some of these factors are environmental like Global warming, loss of habitat, lack of proper vegetation in the surroundings, human intrusion, etc. But other factors include intrusion by other organisms like the infamous Varroa Mites, robber bees, hive beetles, etc. These organisms act as parasites for the colonies and reduce the health in general of the bees in the hive.

Due to all of these factors, constant check-ups need to be performed by the hive-keeper. These manual check-ups disrupt the functioning of the hive if performed repeatedly. Also, these manual check-ups are time consuming and require significant manpower and care by the officials and keepers.

This was our motivation to come up with an automated system that explores honey bee colony health using image data for

- Colony Size, presence and abundance of eggs, larvae, pupae, adults, surface area covered by adult bees.
- Proportion of 3 castes (Numbers of Drones, Workers, Queen)
- Total number of brood cells estimated by measuring the brood area
- Brood pattern consistency
- Indirect demographic characteristics using brood inspection: Proportion of Drone cells, Worker cells, Queen cells
- Presence of disease/insects/mites/fungi/other pathogen
- Productivity (number of cells containing honey, pollen, wax), surface area covered by cells containing honey, pollen, beebread

This semester I completed the task of Counting of honey bees given a frame image of the hive and Classification of honey bees on the basis of subspecies of Apis Mellifera and on the basis of honey bee health. These have huge implications on the honey industry due to the automated nature of honey bee health analysis.

Code can be found on: https://github.com/anvitmangal/IP_Winter-19 for both the tasks.

2 Literature Review

Significant amounts of literature review was important to understand the concepts behind honey bee colonies, health, reasons for the declining health, existing work in health/subspecies classification, scientific and technical terms associated with honey bees. Dr. Swapna Purandare was utmost helpful in making me understand the biological terms and getting me familiarised with the biology component of the literature.

For honey bee counting problem, I looked for papers which tried dense segmentation of humans/objects. Luckily I came through a paper which did dense object tracking in honeybee hives. The paper titled "Towards dense object tracking in a 2D honeybee hive" was published in CVPR 2018 by the OIST Vision research group.

This paper proposes the following: automatically recognize all individuals in a dense group from raw image data. Recognising individual bees involves:

- Segmentation of the individual bees
- Orientation angle of the individual bees

They labelled the data in the following manner: Each bee instance was assigned (x,y,t,alpha) denoting the coordinates of the bee centre, t is 1 if full body visible and 2 if inside a comb cell. Alpha is the angle of the body's axis with the vertical. For labels with t = 1 the regions were ellipse-shaped with semi-minor axis r1 = 20 pixels and semi-major axis r2 = 35 pixels, and rotated by the angle (Fig. 3b). For labels where t = 2 the regions were circular with r = 20 pixels. They used the following Deep Learning methods:

- They integrated the fully convolutional neural network U-Net with a recurrent component object detection in a video sequence.
- To further indicate the head direction on the main body axis, they proposed a loss function approximating individual orientation angle and expand the foreground- background segmentation with object orientation angle estimation.
- The recurrent component of the network leverages the information encoded in the video sequence and improves accuracy, while keeping the network at a fraction of the size of the original U-Net.

- The loss function was defined as: L = wc sin ((alpha-alpha^)/2) where wc is the class weight and alpha, alpha^ are the predicted and labeled orientation angle, respectively.

They achieved the following results: The model could correctly detect 96% of individuals with a location error of 7% of a typical body dimension, and orientation error of 12 deg.

3 Existing Work

3.1 Counting Task

I came across work done by Mat Kelcey which uses a similar approach as the OIST paper. It uses a vanilla U-Net implementation inplace of a more tuned implementation as OIST. A fully convolutional network trained on half resolution patches but run against full resolution images. Encoding is a sequence of 4 3x3 convolutions with stride 2. Decoding is a sequence of nearest neighbours resizes + 3x3 convolution (stride 1) + skip connection from the encoders. Final layer is a 1x1 convolution (stride 1) with sigmoid activation (i.e. binary bee / no bee choice per pixel). After some empirical experiments he chose to only decode back to half the resolution of the input. He did the decoding using a nearest neighbour resize instead of a deconvolution. The network was trained with Adam optimizer.

Following are the results that he got on a few test images using this implementation:

sample actual vs predicted for some test data												
actual	40	19	16	15	13	12	11	10	8	7	6	4
v2 (centroids) predicted	39	19	16	13	13	14	11	8	8	7	6	4
v3 (raw count) predicted	33.1	15.3	12.3	12.5	13.3	10.4	9.3	8.7	6.3	7.1	5.9	4.2

Figure 1: Counting Results Table

3.2 Classification Task

A came across the kernel ran by Dimitry Pukhov for the Kaggle honeybee dataset. He used simple 3 layered convolutional neural networks and achieved the following results:

Classification of subspecies: Accuracy = 0.86

Classification of subspecies: Accuracy = 0.85

4 Datasets

There were majorly three datasets that were used.

4.1 OIST Frame Dataset

This dataset was created as a part of the study Towards dense object tracking in a 2D honeybee hive. It comprises frames and annotations of 2 video recordings of an observation bee hive. The recordings were done at 30 fps and 70 fps, in each of them 360 frames were annotated in a consecutive sequence at 2 fps. The annotation includes positions and orientation angles of all bees in an image within a selected region with high bee density. Annotating was split into tasks on 1024x1024 px windows as shown below:

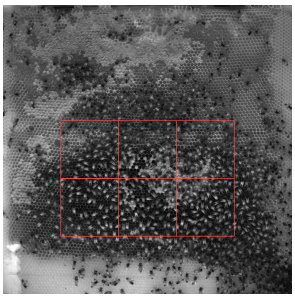

Figure 2: OIST 1

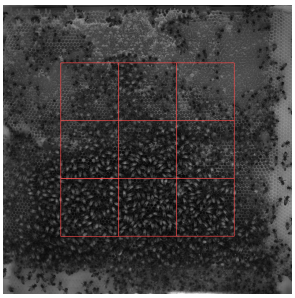

Figure 3: OIST 2

2 object classes were introduced: (1) fully visible bees, (2) abdomens of bees partially hidden inside cells of a honey comb. The orientation angle of objects of class 2 is always 0. The format of annotation files is: offset_x offset_y class position_x position_y angle

4.2 Kaggle BeeImage Dataset: Annotated Honey Bee Images

This dataset contains 5,100+ bee images annotated with location, date, time, subspecies, health condition, caste, and pollen. The original batch of images was extracted from still time-lapse videos of bees. By averaging the frames to

calculate a background image, each frame of the video was subtracted against that background to bring out the bees in the forefront. The bees were then cropped out of the frame so that each image has only one bee. Because each video is accompanied by a form with information about the bees and hive, the labeling process is semi-automated. Each video results in differing image crop quality levels. This dataset will be updated as more videos and data become available. -1 means the information is coming soon.

Figure 4: Kaggle Bee Image 1

Figure 5: Kaggle Bee Image 2

4.3 Apis Mellifera Bee frames dataset

This dataset was procured by Dr. Swapana Purandare. Videos were shot on a hand-held device of about 1 hour in length. I then extracted out images from these videos at 1 frame per 10 frames in the video so as to get a different configuration of honey bees in each frame. This dataset has 4000 images in total.

5 Visits

We visited the Indian Council of Agricultural Research, Delhi to talk to the research heads there regarding the procurement of datasets for our research. They had organised a demo for us to make us understand how bee colonies work.

Figure 6: Procured Image 1

Figure 7: Procured Image 2

6 Methods

Two deep learning models that are very famous in the field of computer vision were used for my experiments. U-Net was used for the counting problem and ResNet-18 for the classification task.

6.1 UNet

The UNET was developed by Olaf Ronneberger et al. for Bio Medical Image Segmentation. The architecture contains two paths. First path is the contraction path (also called as the encoder) which is used to capture the context in the image. The encoder is just a traditional stack of convolutional and max pooling layers. The second path is the symmetric expanding path (also called as the decoder) which is used to enable precise localization using transposed convolutions. Thus it is an end-to-end fully convolutional network (FCN), i.e. it only contains Convolutional layers and does not contain any Dense layer because of which it can accept image of any size. In the original paper, the UNET is described as follows:

On a high level, we have the following relationship: Input (128x128x1) =¿ Encoder =¿(8x8x256) =¿ Decoder =¿Ouput (128x128x1)

6.2 Resnet

After the celebrated victory of AlexNet at the LSVRC2012 classification contest, deep Residual Network was arguably the most groundbreaking work in the computer vision/deep learning community in the last few years. ResNet makes it possible to train up to hundreds or even thousands of layers and still achieves

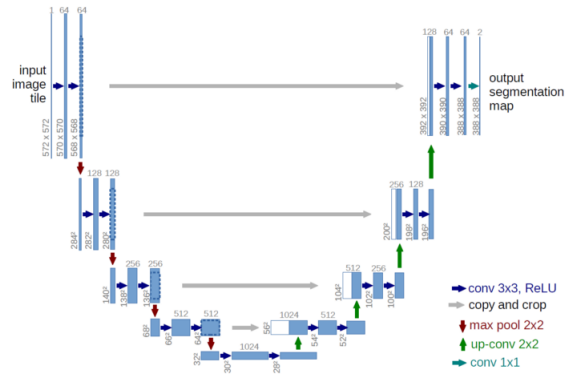

Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Figure 8: UNet Architecture

compelling performance.

The core idea of ResNet is introducing a so-called "identity shortcut connection" that skips one or more layers, as shown in the following figure:

The authors argued that stacking layers shouldn't degrade the network performance, because we could simply stack identity mappings (layer that doesn't do anything) upon the current network, and the resulting architecture would perform the same. This indicates that the deeper model should not produce a training error higher than its shallower counterparts. They hypothesize that letting the stacked layers fit a residual mapping is easier than letting them directly fit the desired underlying mapping. And the residual block above explicitly allows it to do precisely that.

7 Training Procedure

7.1 Counting task

For the counting problem, I used the github library by the OIST research group: https://github.com/oist/DenseObjectDetection. Using this I trained the U-Net model on the OIST frame dataset after performing a train-test split.

Following are some of the hyperparameters I used: Batch-size = 8 Number of steps = 5000 Background/Foreground Weight = 0.9

The code is implemented in tensorflow and the tf Estimators were used to train/test the model.

I then tested the model on the test set of 50 images.

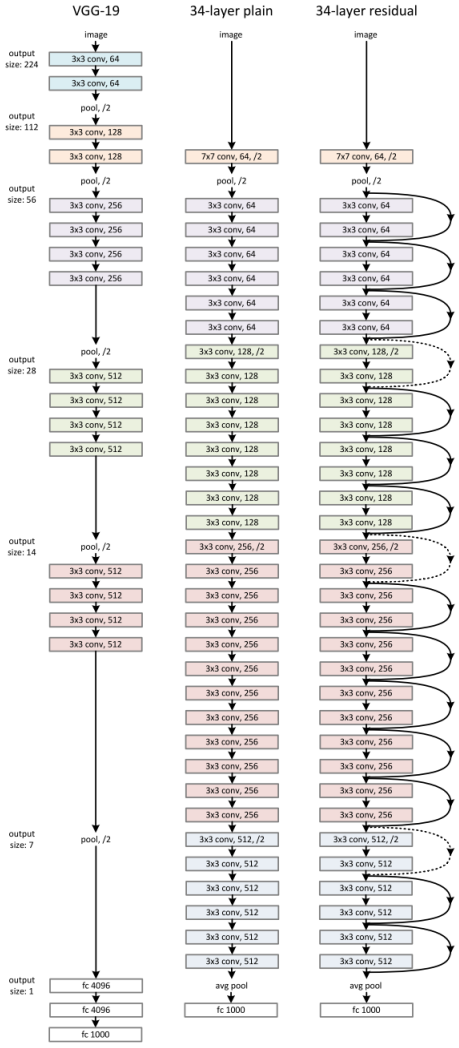

Figure 9: Resnet Architecture

7.2 Classification task

For both the tasks, I used Resnet-18 due to its success in image classification task and at the same time having a reasonable amount of free parameters to train, owing to the comparably small training dataset in hand: Kaggle Honeybee dataset with 5000 images.

I used the pytorch imagenet github library for training the resnet-18 model. I used the following hyper-parameter configuration:

Epochs = 90 Batch-size = 64 Learning rate = 0.1 Momentum = 0.9 Weight-

decay = 1e-4

Following are the validation loss and accuracy curves:

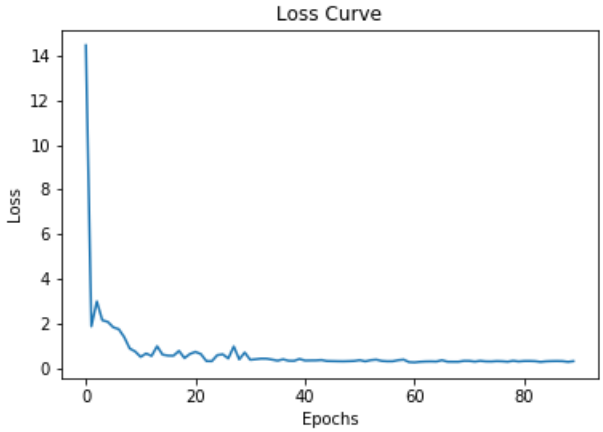

Figure 10: Loss vs Epoch Curve

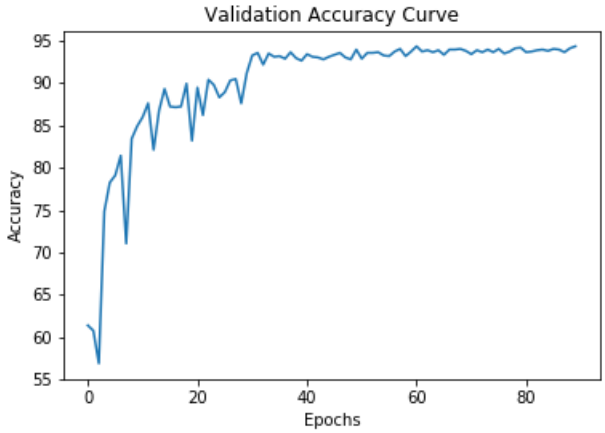

Figure 11: Accuracy vs Epoch Curve

Later I tried DenseNet due to its similar performance in Object detection tasks. DenseNet improved the results by a small margin.

8 Results

8.1 Classification Task

- Using Resnet:
  - Subspecies classification: 94.28 % , baseline: 86.5%
  - Health Classification: 95.75 % , baseline: 84.9%
- Using DenseNet:
  - Subspecies classification: 97.77 % , baseline: 86.5%
  - Health Classification: 98.40 % , baseline: 84.9%

8.2 Counting Task

Mean square error: 246 Mean absolute error: 16.64 Standard Deviation of absolute error: 11.52 Graph for predicted and ground truth counts:

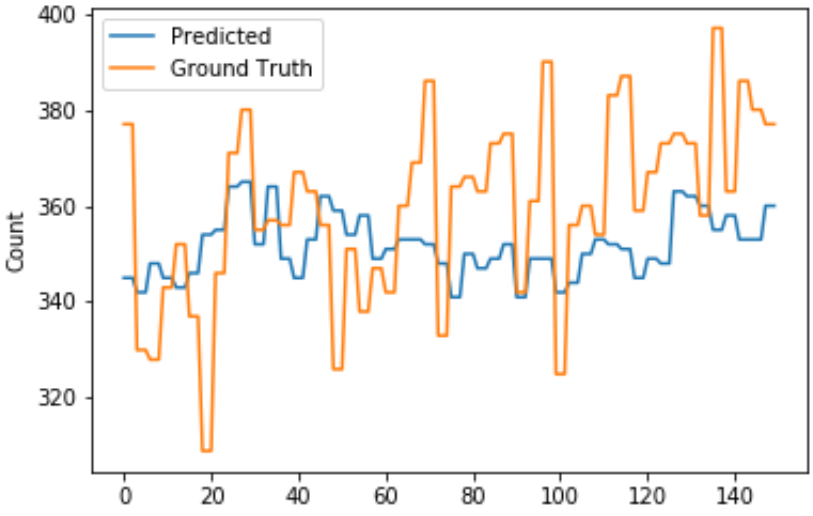

Figure 12: Counting: Ground Truth and Predictions

9 Challenges

- Procuring the dataset was one of the major issues. We had to put significant efforts in finding suitable datasets for our purposes.

- Due to the significant literature on honeybees and learning the jargon associated to bees and their colonies was a major task as reading research papers was becoming difficult.

10 Future Work

- Semi-supervised labelling of our procured dataset on Apis-Mellifera bees.
- Training of the U-Net model on this dataset to generalise to the many type of configurations in our dataset.
- Procurement of a dataset on Apis-Cerana, and repeating the above procedure on this dataset.
- Unsupervised domain adaptation can be used to use the unlabelled procured dataset to train the model, and generalise on indian honey bees as well.

References

[1] Towards dense object tracking in a 2D honeybee hive
https://arxiv.org/abs/1712.08324

[2] Counting Bees using UNet
`http://matpalm.com/blog/counting_bees/`

[3] Honey-bee Annotated Dataset
`https://www.kaggle.com/jenny18/honey-bee-annotated-images`

[4] Kaggle Kernel for Honey-Bee annotated dataset
`https://www.kaggle.com/dmitrypukhov/honey-bee-health-detection-with-cnn`

[5] U-Net: Convolutional Networks for Biomedical Image Segmentation
`https://arxiv.org/abs/1505.04597`

[6] Deep Residual Learning for Image Recognition
`https://arxiv.org/abs/1512.03385`