8-2019

Design and Evaluation of Convolutional Networks for Video Analysis of Bee Traffic

Prateek Vats
Utah State University

DESIGN AND EVALUATION OF CONVOLUTIONAL NETWORKS FOR VIDEO ANALYSIS OF BEE TRAFFIC

by

Prateek Vats

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Computer Science

Approved:

Vladimir Kulyukin, Ph.D. Major Professor	Nicholas Flann, Ph.D. Committee Member
Xiaojun Qi, Ph.D. Committee Member	Richard S. Inouye, Ph.D. Vice Provost for Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2019

ABSTRACT

Design and Evaluation of Convolutional Networks for Video Analysis of Bee Traffic

by

Prateek Vats, Master of Science

Utah State University, 2019

Major Professor: Vladimir Kulyukin, Ph.D.
Department: Computer Science

This thesis presents a novel approach to estimate bee traffic levels in a Langstroth beehive by employing Deep Learning and computer vision techniques to recognize bees. Estimating bee traffic levels will further help in analyzing bee colony behavior and health. Various Convolutional Neural Networks (ConvNet) models were designed, trained, and validated to find the best model for bee recognition. The best model is then integrated with a motion detection algorithm running on a Raspberry Pi 3 computer as a part of BeePi, a multi-sensor electronic beehive monitoring system, that was designed in 2014 and has been iteratively developed ever since.

We compared the bee detection accuracy of ConvNets with traditional machine learning models and found ConvNets to be superior. Best performing models were then used to estimate bee traffic levels, aligned with real Beehive inspections. This was further extended to find a correlation between the Beehive data points extracted from inspections and bee traffic levels.

(59 pages)

PUBLIC ABSTRACT

Design and Evaluation of Convolutional Networks for Video Analysis of Bee Traffic

Prateek Vats

Colony Collapse Disorder (CCD) has been a major threat to bee colonies around the world which affects vital human food crop pollination. The decline in bee population can have tragic consequences, for humans as well as the bees and the ecosystem. Bee health has been a cause of urgent concern for farmers and scientists around the world for at least a decade but a specific cause for the phenomenon has yet to be conclusively identified.

A normal hive inspection can be very disruptive for the bee colony, as the hive needs to be disassembled to visually assess hive health from the inside by collecting larvae and egg data. This work uses Machine Learning and Computer Vision methodologies to develop techniques to monitor hive health without disrupting the bee colony residing in the hive. Bee traffic refers to the number of bees moving in a given area in front of the hive over a given period of time. Bee traffic is related to forager traffic. Forager traffic is the number of bees moving out of the beehive. Forager traffic is a crucial factor in determining and monitoring food availability, food demand, colony age structure, the impact of pesticides, etc. on beehives. This work focuses on estimating bee traffic levels in a given hive and associate this information with data collected through manual beehive inspections.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Vladimir Kulyukin, who always had his office doors open for his students. He guided me all through my explorations with the best recommendations pushing me closer to my goals. His patience and immense knowledge greatly inspired me and contributed to the accomplishment of this research.

I would also like to express my appreciation to the rest of my committee members, Dr. Xiaojun Qi, and Dr. Nicholas Flann, for their continuous support and for providing their insightful remarks.

I also want to thank my lab mates for participating in insightful and stimulating discussions, that helped me gain more perspective and come up with ideas that helped me in this research.

Prateek Vats

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

The Western honey bee (Apis mellifera L.) is the worlds premier pollinator species. Due to soaring demands for fruits and especially nuts, their importance as a pollinator has increased as well. That represents almost 100 crop species, making up one-third of the average diet. In the United States, honey bees pollinate an estimated $15 billion of crops each year [3].

Apiculture has been on the decline [4] in both the USA and Europe over recent decades. It is, therefore, crucial to make beekeeping a more attractive hobby and a less laborious profession, in order to encourage local apiculture and pollination. Sudden losses of honey bee colonies have occurred, and have received considerable public attention. Colony Collapse Disorder (CCD) in the USA has attracted great attention, and scientists here and in Europe are working hard to provide explanations for these extensive colony losses. Disruption of the honey bee supply causes prices for domestically grown nuts, fruits, and vegetables to go up.

These factors have led bee health to be a cause of urgent concern for scientists and farmers around the world for at least a decade and a lot of work has been done in past for data collection and analysis for its research. One of the important variables in order to monitor food availability, food demand, colony age structure, the impact of pesticides, etc. in bee-hives is forager traffic [5]. Forager activity is an important variable to monitor when evaluating the impact of pesticides on honey bee colony health [6]. Monitoring forger traffic will lead to improved remote monitoring of general hive status and improved real-time detection of the impact of pests, diseases, pesticide exposure, and other hive management problems. Since forager traffic can be affected by food availability, food demand, and

colony age structure. Thus, sudden changes in that traffic may indicate acute changes on the colony level. Forager activity is described in terms of the number of bees entering and/or exiting the hive over a given time period, data can be collected, if need be, without the use of equipment more sophisticated than an observer and a stopwatch. The use of human observation, while likely to be accurate, clearly limits due to fatigue and the amount of time that the hive can be observed.

The importance of estimating the health of bee-hives demands research, data collection and beehive setups over the span of several years. EBM Systems can help in automating the collection effort of a large amount of information to assess hive health without invasive and disruptive colony inspections. With more upcoming and previous research work on such systems, EBMs have gained more popularity over the years. Since the usage of object detection methodologies plays a prime role in these systems, a good amount of high-quality work has been done in the field of object detection using machine learning.

The importance of estimating the health of bee colonies demands research and advancement over the span of several years in this area. EBM systems help in automating the task of collecting a large amount of useful information on behavior without invasive and disruptive colony inspections. As EBMs gained popularity, it became important to focus on the design of EBMs as well as incorporating modern software solutions to aid and better utilize them. Related work has been described below.

1.2 Related work

The significance of beehive monitoring and estimating its health demands research and advancement over the span of several years, making it imperative to concentrate on EBM designs as well as incorporating new methodologies to improve their utilization. EBM frameworks help in robotizing the assignment of gathering vast measure of valuable data without obtrusive and intrusive manual inspections.

Some work involved techniques based on computer vision to solve the bee motion counting effort which further contributes to estimating overall bee traffic. The first methodology used contour detection. An image is binarized and a list of contours through connected

components is computed [7]. The list of contours is further processed to estimate bee motion count. The second methodology used color thresholding to determine if a particular pixel is a landing pad or bee. Using this, an overall bee pixel area is computed and divided by an estimated single bee pixel area. This gives an estimate of the number of bees in an image [8].

With Deep Learning, having made a significant advancement, one of the methods used 32x32 pixel images that were labeled from frames extracted from videos collected from BeePis installed in the summer of 2016. This work explores Deep Learning methodologies and presents much better performance than the previous methods [9].

The task of image classification using machines has been quite challenging in terms of their performance comparison with humans, because of the differences in their representation. Humans are capable of detecting visual patterns and learn to classify images much faster than machines. Images although visual to us, are represented in pixel values as opposed to raw binary data in machines. It has been challenging to find patterns in raw binary data to help with classification problems. There has been significant advancement in that field since the last decade. Convolution layers, a state of the art deep learning technique, has significantly improved a machine's image classification capability. A considerable amount of work has been done in diving deep into the working of Convolution Networks. It presents novel visualization techniques to provide more insight into the functioning of feature layers and operations of classification [10].

General descriptors extracted from CNNs have been proved to be powerful by one of such works where an existing network was used to extract features, and great results were seen [11].

With the coining of the term Deep Learning, there has been some work done on finding the answer to the question. How deep can we go? Karen et al. [12] presented their findings by thoroughly evaluating networks of increasing depth with small convolution filters. They went as deep as 19 convolution layers. The work showed to have a better generalization over other state-of-the-art networks.

Another interesting work called for the usage of CNNs for region proposals termed as R-CNNs, which basically comes up with segmented areas in an image that contain CNN features. An improvement to this was also published by Dumitru et al. [**?**], which solved the problem of detecting multiple instances of the same object in an image. Previous works like R-CNNs failed to do so. The model proposed is able to handle detection of multiple instances of the same class.

Although Deep Neural networks, claim to classify as well as humans, there has been some work done to prove that Deep Neural networks can easily be fooled. Convolutional neural networks trained to perform well on ImageNet or MNIST are taken and images that evolutionary algorithms or gradient ascent that DNNs labeled with high confidence are found. These images are totally unrecognizable to human eyes that DNNs believe to be familiar. The results of this work present interesting differences between human vision and classifiers created through DNNs [13].

With more work being done on image classification, the problem of detecting objects became relevant. One of such works explores the problem of localization along with classification. It presents object detection as a regression problem to calculate object bounding box masks [14].

Although object detection has gained momentum, the task of counting similar objects has been a challenging task. The count-ception algorithm [15] explored a method where a regression network predicts a count of the objects being detected in an image. This is done by processing the image in a fully convolutional way. Redundant counting is done instead of predicting a density map to avoid errors. CentroidNet presented by K.Dijkstra et al. [16] also presented a novel method of utilizing Fully Convolutional Neural Network on a field of vectors extracted from labeled data and combined with a segmentation map for accurate object centroid detection.

1.3 Current work

In this thesis, we explore techniques using machine learning and computer vision to estimate bee traffic levels. ConvNets have proved to perform well on image classification.

We use computer vision based techniques along with deep learning to collect bee traffic information to assess a hive's health. This thesis has been organized as follows; Chapter 2 discusses ConvNets and traditional machine learning methods in detail, focusing on the mathematics behind them. Chapter 3 explains the process of how the data was collected, labelled, processed, and analyzed using MANOVA. It also describes the overall algorithm used to collect bee motion count data through videos collected from each hive. Chapter 4 goes into explaining in detail - the experiments conducted, their results, and our analysis. Chapter 5 goes through conclusions drawn from the experiments and potential future work.

CHAPTER 2

CLASSIFICATION MODELS

Over last few years, with significant push in research in the field of Deep Learning, numerous methods have appeared to outperform the previous state of the art machine learning (ML) methods in several fields, with computer vision noticeably standing out the most. Conventional ML techniques are limited in their ability to process raw data from images and perform relevant feature extraction. For a long period of time, ML techniques required careful feature engineering or considerable domain knowledge to design a feature extractor that transformed raw data into a relevant representation of a feature vector from which a learning system such as a classifier could detect or classify patterns in the input. However, the performance of ML algorithms suffers substantially when our data is masked under irrelevant features. The the goal behind DL is to learn the features from data without any supervision; here the algorithms will do the feature engineering to provide deep neural network models with meaningful features to most accurately produce the desired output.

The next sections provide detailed explanations of traditional machine learning models such as SVMs, Random Forests, and deep learning models such as Convolution Neural Networks and its components.

2.1 Traditional Machine Learning

Linear Regression is a supervised machine learning algorithm where the predicted output is continuous and has a constant slope. Its used to predict values within a continuous range, rather than trying to classify them into categories. There are two main types-

Simple linear regression uses a traditional slope-intercept form, where m and b are the variables our algorithm will try to learn to produce the most accurate predictions. x represents our input data and y represents our prediction.

$y = mx + b$

A more complex, multi-variable linear equation might look like this, where w represents the coefficients, or weights, our model will try to learn.

$f(x, y, z) = w_1x + w_2y + w_3z$

SVM

Support vector machines (SVMs) are a set of related supervised learning methods that analyze data and recognize patterns, used for classification (machine learning)—classification and regression analysis. The original SVM algorithm was invented by Vladimir Vapnik and the current standard incarnation (soft margin) was proposed by Corinna Cortes and Vladimir Vapnik [17] . The standard SVM is a non-probabilistic binary classifier—binary linear classifier, i.e. it predicts, for each given input, which of two possible classes the input is a member of. Since an SVM is a classifier, then given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category or the other. Intuitively, an SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

More formally, a support vector machine constructs a hyperplane or set of hyperplanes in a high or infinite dimensional space, which can be used for classification, regression or other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

Random Forests

A decision tree is a flowchart-like graphical depiction of a decision and every possible outcome to make that decision. On the other hand, Random Forest is an ensemble method which makes use of decision trees. In this method, multiple models are built using a subset of features and trained on different training sets which allow trees to decorrelate from each

other. Each model is then used individually to predict an independent result, after which a vote is taken to predict the best class in case of classification problems. Decision trees are more prone to suffer from high variance or high bias whereas Random Forest takes care of the bias-variance tradeoff by finding out a balance between the two extreme sides.

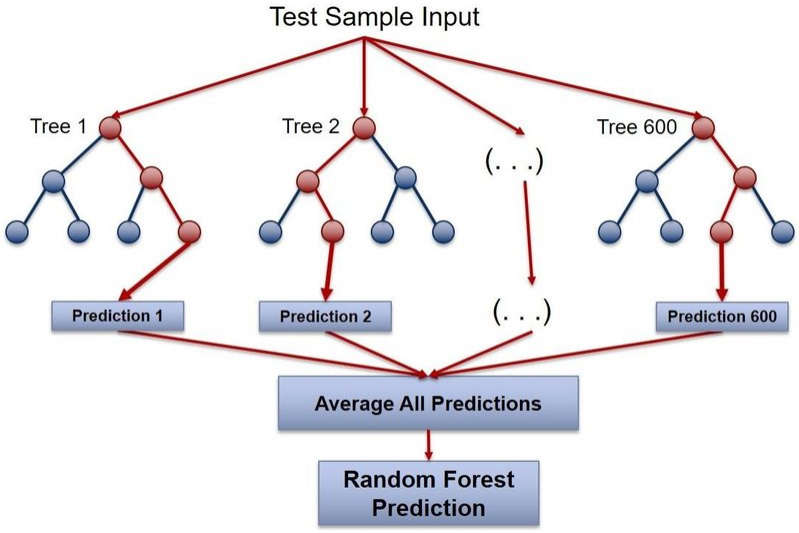

Fig. 2.1: Visualization of Random Forest [1]

2.2 Deep Learning

Convolution Neural Network

ConvNets are a derivative of standard MLP neural networks optimized for two-dimensional pattern recognition problems. Instead of using fully connected hidden layers as described in the preceding section, the ConvNet introduces a special network structure, which consists of convolution and sub-sampling layers. Feature maps generated by convolution layers, contain neurons that take their synaptic inputs from a local receptive field. The weights of neurons within the same feature map are shared. It allows to have replicated units sharing the same configuration, thereby features can be detected regardless of their position in the visual field. Moreover, the fact that weights are shared increases learning efficiency by reducing

the number of parameters being learned. In order to have a data reduction, a sub-sampling operation called pooling is performed. This data reduction operation is applied to the predecessor convolution result by a local averaging over a predefined window. It partitions the input image into a set of non-overlapping windows and then for each sub-region outputs the maximum value. This step is important because it helps to eliminate non- maximal values. The output layer ensures the classification of the input image. In this layer all neurons are fully connected and have a unique set of weights so they can detect complex features and perform classification. Apart from input and output layer, what makes ConvNets different from MLPs are Convolutional Layer, Pooling Layer, and Fully-Connected Layer. We will describe each layer and its working in details in below sections.

Convolution Layer

The main task of the convolutional layer is to detect local conjunctions of features from the previous layer and mapping their appearance to a feature map. As a result of convolution in neuronal networks, the image is split into perceptrons, creating local receptive fields and finally compressing the perceptrons in feature maps of size m2 m3. Thus, this map stores the information where the feature occurs in the image and how well it corresponds to the filter. Hence, each filter is trained spatial in regard to the position in the volume it is applied to.

In each layer, there is a bank of m_1 filters. The number of how many filters are applied in one stage is equivalent to the depth of the volume of output feature maps. Each filter detects a particular feature at every location on the input. The output $Y_i^{(l)}$ of layer l consists of $m_1^{(l)}$ feature maps of size $m_2^{(l)} m_3^{(l)}$. The i^{th} feature map, denoted $Y_i^{(l)}$, is computed as

$$Y_i^{(l)} = B_i^{(l)} + \Sigma_{j=1}^{m_1^{(l-1)}} K_{i,j}^{(l)} * Y_j^{(l-1)} \tag{2.1}$$

The result of staging these convolutional layers in conjunction with the following layers is that the information of the image is classified like in vision. That means that the pixels are assembled into edglets, edglets into motifs, motifs into parts, parts into objects, and

objects into scenes

Activation functions

Activation functions are really important for a Artificial Neural Network to learn complicated and non-linear complex functional mappings between the inputs and target variable. They introduce non-linear properties to the Network. They the output of that node, or "neuron," given an input or set of inputs. This output is then used as input for the next node and so on until a desired solution to the original problem is found.

In my experiments there has been a use of several activation functions such as-

1. Sigmoid function

   $f(x) = \frac{1}{1+e^{-x}}$

2. TanH function

   $f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$

3. Softmax function

   $f(x) = \frac{e^{x_i}}{\Sigma_{j=1}^{J} e^{x_j}}$ for $i = 1, ..., J$

4. ReLu

   $$f(x) = \begin{cases} 0, & \text{for } x < 0. \\ x, & \text{for } x >= 0. \end{cases}$$

Pooling

The pooling or downsampling layer is responsible for reducing the spacial size of the activa- tion maps. In general, they are used after multiple stages of other layers (i.e. convolutional and non-linearity layers) in order to reduce the computational requirements progressively through the network as well as minimizing the likelihood of overfitting. The pooling layer l has two hyperparameters, the spatial extent of the filter F (l) and the stride S (l) . It takes an input volume of size $m_1^{(l-1)} x m_2^{(l-1)} x m_2^{(l-1)}$ and provides the volume of size $m_1^{(l)}$ x $m_2^{(l)}$ x $m_2^{(l)}$ where;

$$m_1^{(l)} = m_1^{(l-1)}$$
$$m_2^{(l)} = (m_2^{(l-1)} - F^{(l)})/S^{(l)} + 1$$
$$m_3^{(l)} = (m_3^{(l-1)} - F^{(l)})/S^{(l)} + 1$$

The key concept of the pooling layer is to provide translational invariance since particularly in image recognition tasks, the feature detection is more important compared to the features exact location. Therefore the pooling operation aims to preserve the detected features in a smaller representation and does so by discarding less significant data at the cost of spatial resolution. The pooling layer operates by defining a window of size $F^{(l)}$ x $F^{(l)}$ and reducing the data within this window to a single value. The window is moved by $S^{(l)}$ positions after each operation similarly to the convolutional layer and the reduction is repeated at each position of the window until the entire activation volume is spatially reduced.

Fully Connected Layer

The fully connected layers in a ConvNet are practically a multilayer perceptron (generally a two or three layer MLP) that aims to map the $m_1(l1)$ x $m_2(l1)$ x $m_3(l1)$ activation volume from the combination of previous different layers into a class probability distribution. Thus, the output layer of the multilayer perceptron will have $m_1(li)$ outputs, i.e. output neurons where i denotes the number of layers in the multilayer perceptron.

The key difference from a standard multilayer perceptron is the input layer where instead of a vector, an activation volume is taken as the input. As a result the fully connected layer is defined as:

If l-1 is a fully connected layer

$$Y_i^{(l)} = f(Z_i^{(l)}) = \sum_{j=1}^{m_1^{(l-1)}} w_i^{(l)} y_i^{(l-1)}$$

otherwise,

$$Y_i^{(l)} = f(Z_i^{(l)}) = \sum_{j=1}^{m_1^{(l-1)}} \sum_{r=1}^{m_2^{(l-1)}} \sum_{s=1}^{m_3^{(l-1)}} w_{i,j,r,s}^{(l)} (Y_i^{(l-1)})_{r,s}$$

The goal of the complete fully connected structure is to tune the weight parameters $w_{i,j}^{(l)}$ or $w_{i,j,r,s}$ to create a stochastic likelihood representation of each class based on the activation maps generated by the concatenation of convolutional, non-linearity, rectification and pooling layers. Individual fully connected layers operate identically to the layers of the multilayer perceptron with the only exception being the input layer.

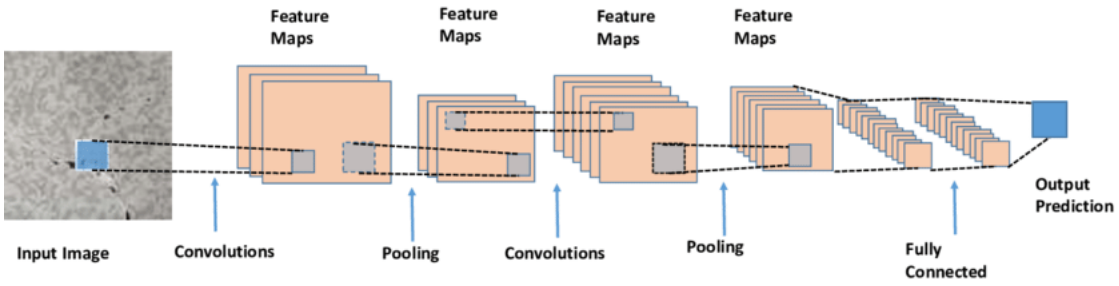

Fig. 2.2: Convolutional Neural Network Architecture [2]

CHAPTER 3

DATASET

3.1 Overview

This section describes in detail the process of collecting data to train models to detect bees in videos. The process has been divided into following steps - data collection, data labeling, MANOVA analysis on data, and collecting bee motion count data. Subsequent sections describe each stage in detail.

3.2 Data Collection using BeePi

A BeePi is a computer running on Raspbian, a Linux flavor, that runs on a Raspberry Pi. A key target of the BeePi configuration is reproducibility and reusability: different analysts and professionals ought to have the capacity to recreate our outcomes at least expense and time responsibilities. Every BeePi comprises of a Raspberry Pi computer, a smaller than usual camera, a sunlight based board, a temperature sensor, a battery, an equipment clock, and a sun-based charge controller. Figure 3.1 demonstrates an image of the BeePi framework. This collection of hardware is placed in a wooden box which is placed on top of all the boxes that contain hive frames. BeePi is intended for Langstroth hives [18] utilized by numerous beekeepers around the world. Four BeePi EBM systems were gathered and setup at two Northern Utah apiaries to gather 27.0 GB of sound, temperature, and video information in various climate conditions.

For seamless integration with the BeePi framework, we worked with the Raspberry Pi 3 computer that is a part of BeePi. The Pi-3 has 1GB of RAM and 4 cores that are more than equipped for handling recordings with ConvNets. The information utilized for preparing and testing the Neural Network models is extracted from the recordings gathered by the solar-powered, EBM System, called BeePi.

4 BeePi's were setup in the summer of 2018. These were named - 4.5, 4.7, 4.8 and 4.10. The Figures 3.1, 3.3, 3.2 illustrate the overall setup of these systems.

Fig. 3.1: The Overall setup of a BeePi

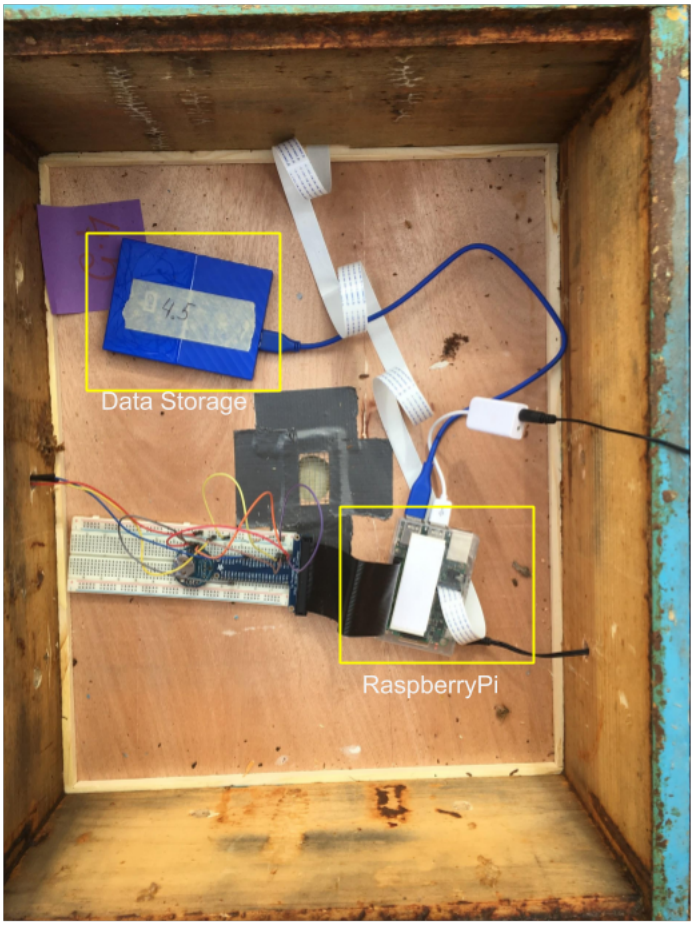

Fig. 3.2: Hardware components of BeePi inside a Langstroth super

Fig. 3.3: Pi Camera facing the landing pad

The whole setup consists of a box, which we refer to as "supers" and can contain around 5 frames. Here frames are rectangular meshes on which the bees will build their hives, reproduce and store the honey. As the bee population increases, we need to add more of these boxes, to accommodate further bee population growth in the future.

The Raspberry Pi camera mounted over the top-most super and facing the landing pad, as seen in the image above. It captures bee movement over the landing pad. It is expected that as the population increases, the movement shall increase.

As we add supers, the distance of the pi camera increases, which means, if we capture the videos at the same resolution, the bees would appear smaller when we have two supers compared to when we just have a single box. Fig 3.5 and Fig 3.5 show the comparison of a Super 1 video vs Super 2 video.

Fig. 3.4: Super 1 video snapshot

Fig. 3.5: Super 2 video snapshot

Super 1 videos captured for all the hives have been recorded from 05/05/2018 until approximately 06/09/2018 and Super 2 videos captured for all the hives have been recorded from 06/09/2018 until approximately 07/07/2018. Every day from 8 am to 9 pm, the videos were captured every 15 minutes for 30 seconds at a resolution of 1920x1080. To count bees in these videos, we needed to train Neural Network to classify small chunk in a frame from these videos as a bee or no-bee.

In total, there are 5,509 super 1 videos and 5,460 super 2 videos. To fulfill that purpose, we used background subtraction to extract out the chunks from videos. Since the bees keep moving and it is expected that the background will remain static, we could extract images from frames of the videos to manually label as a bee or a no-bee. Since the size of the bees varies depending on the number of boxes. After careful analysis, we chose a crop size of 150x150 for super 1 videos and 90x90 for super 2 videos. We extracted in total 58,484 images from super 1 videos and 57,671 images from super 2 videos. We will be calling these datasets super 1 and super 2 data throughout this article.

3.3 Data Labelling

We labeled in a total of 116,155 images from super 1 and super 2 data combined. These images were labeled out of approximately 100 videos(50 videos each from super 1 and super 2 videos) in total out of the 10,969 videos captured. We labeled them as either BEE or NO-BEE. In order to test generalization of our models we arranged all this data in two ways:

Fig. 3.6: A sample of super 1 images from BEE2 (BEE2_1S); the first four rows include images classified as BEE; the last three rows consist of images classified as NO-BEE.

Fig. 3.7: A sample of super 2 images from BEE2 (BEE2_2S); the first four rows include images classified as BEE; the last three rows consist of images classified as NO-BEE.

3.3.1 Generalized Dataset

In this dataset we labelled all the training/testing data from 4.5 and 4.7, and the validation data from 4.8 and 4.10. For super 1 we had 47,491 images for training/testing and 10,991 images for validation. For super 2 we had 41,446 images for training/testing and 16,222 for validation.

Super 1 training consists of 11,322 BEE images and 36,167 NO-BEE images. super 1 validation consists of 8,307 BEE images and 2,682 NO-BEE images. super 2 training consists of 17,316 BEE images and 24,310 NO-BEE images. Super 2 validation consists of 6,829 BEE images and 9,391 NO-BEE images. The purpose of this dataset was to test how generalized can our models be.

3.3.2 Combined Dataset

In this dataset we combined all the images labelled from 4.5, 4.7, 4.8 and 4.10, and

split them into training/testing and validation. The split was approximately 85% for training/testing and 15% for validation. For super 1 we had 52,214 images for training/testing and 6,268 images for validation. For super 2 we had 51,667 images for training/testing and 6,002 for validation.

Super 1 training consists of 11,322 BEE images and 36,167 NO-BEE images. super 1 validation consists of 8,307 BEE images and 2,682 NO-BEE images. super 2 training consists of 21,146 BEE images and 27,739 NO-BEE images. super 2 validation consists of 3000 BEE images and 6,002 NO-BEE images.

3.4 Bee motion count data

We detect motions in the videos collected from the BeePis installed in each hive, and use our best classifier to categorise them as either BEE or NO-BEE. We segmented a day into 4 time periods - Morning, Noon, Afternoon and Evening. We extracted 2 videos from each time periods, getting a total of 8 videos per day. The 4 time periods fall under the following time ranges-

Morning : 8 am to 11.59 am

Noon : 12 pm to 2.59 pm

Afternoon: 3 pm to 5.59 pm

Evening: 6 pm to 8.59 pm

Number of bees in every frame of the video is counted and stored in a csv format.We perform this on videos collected during the period of 05/05/2018 to 07/07/2018 for both super 1 and super 2 data. For each hive(both super 1 and super 2) following were number of videos this was performed on-

Hive 4.5: 518 videos

Hive 4.7: 546 videos

Hive 4.8: 488 videos

Hive 4.10: 688 videos

Using the 8 videos extracted for each day, we detect BEEs in all these videos and that gives us a bee motion count over different hours of the day. E.g. from the videos of 4/29/2018

we have bee motion counts during the following times - 9:00,10:00,13:00,15:00,18:00,17:00. These bee motion counts are then used to find an average overall bee motion count of the day. This helped us get a high level view of the whole time period for each hive. However, due to unforeseen hardware failures,there are missing detected bee motion count data points for particular dates introducing short discrepancies that can be seen in the Figures - 4.5,4.6,4.7,4.8

3.5 ANOVA/MANOVA

Analysis of variance (ANOVA) [19] is a collection of statistical models and their associated estimation procedures (such as the "variation" among and between groups) used to analyze the differences among group means in a sample. In the ANOVA setting, the observed variance in a particular variable is partitioned into components attributable to different sources of variation. In its simplest form, ANOVA provides a statistical test of whether the population means of several groups are equal, and therefore generalizes the t-test to more than two groups. ANOVA is useful for comparing (testing) three or more group means for statistical significance. It is carried out on a factor - which is a variable common between sets of groups using which we will be gauging if the groups are significantly different or not.

Multivariate analysis of variance (MANOVA) [20] is similar to ANOVA but instead of just one dependent variable. It is used where there are two or more dependent variables. Since an image needs to be represented by a set of multiple variables, we chose to perform MANOVA on different types of datasets.

In our case, we will be extracting out single-valued image features using GLCM TEXTURE [21] implemented as a part of the SKIMAGE library. The single-valued features we will be extracting are contrast,, and energy. In Grey Level Co-occurrence Matrix or GLCM, the texture of the image is calculated by moving a window over the whole image with a specific size and stride of 1 pixel. For contrast, the following formula is used to come up

with a contrast number for a given window-

$$\sum_{i,j=0}^{N-1} P(i-j)^2$$

For homogeniety, the following formula is used-

$$\sum_{i,j=0}^{N-1} \frac{P_{i,j}}{1+(i-j)^2}$$

For energy,a term Angular Second Moment(ASM) is calculated using the following formula-

$$\sum_{i,j=0}^{N-1} P_{i,j}^2$$

The square root of ASM is used to find texture measure, also called Energy using the following formula-

$$Energy = \sqrt{ASM}$$

We performed MANOVA analysis by extracting these three single valued factors for each image belonging to a group to compare how different each group's images are with respect to each other. To compare how significantly different each hive's images are with respect to time periods. We will be using 4 hives as independent variables and dependent variables as contrast,energy and homogeniety. The time periods that we refer to here are Morning, Noon, Afternoon, and Evening.

We extracted 375 images per hive per time period and performed this analysis on three dependent variables extracted as a feature of each image - Contrast, Energy and Homogeneity. We perform this analysis for 4 different time periods - Morning, Noon, Afternoon and Evening images. In MANOVA analysis, P-FACTOR determines if the groups are significantly different or not. In our case, we want to prove that they are significantly different. The P-FACTOR for each time period has been presented in the table below. For the time period images to be significantly different, it has to be < 0.05. We will be presenting the results of this analysis in the sub-sections further.

3.5.1 MANOVA Analysis

The results of this analysis can be seen in Table 3.1 and 3.2. The P-FACTOR is much less than 0.05 for most of the factors which means for morning,noon,afternoon and evening images in super 1 as well as super 2 images our dataset is significantly different, which means there is high variation in data. With this information we can ascertain that we don't have images that look similar and we will be training our models on highly variable images.

Table 3.1: MANOVA analysis of different time periods from super 1 images

Time Period	Pillai Coefficient	F-VALUE	P-FACTOR
Morning	0.089051	260.55	$< 2.2e^{-16}$
Noon	0.089051	260.55	$< 2.2e^{-16}$
Afternoon	0.089051	260.55	$< 2.2e^{-16}$
Evening	0.089051	260.55	$< 2.2e^{-16}$

Table 3.2: MANOVA analysis of different time periods from super 2 images

Time Period	Pillai Coefficient	F-VALUE	P-FACTOR
Morning	0.16858	216.01	$< 2.2e^{-16}$
Noon	0.16858	216.01	$< 2.2e^{-16}$
Afternoon	0.16858	216.01	$< 2.2e^{-16}$
Evening	0.16858	216.01	$< 2.2e^{-16}$

3.5.2 MANOVA - Generalized Dataset - Training vs Testing vs Validation

The MANOVA analysis of super 1 gave the Pillai coefficient of 0.28849, the approx F value of 168.33, and $Pr(> F) < 2.2e - 16$. Since P-value is less than 0.05, we can conclude that our training, testing and validation images from super 1 dataset are significantly different from each other. The MANOVA analysis of super 2 gave the Pillai coefficient of 0.067891, the approx F value of 35.091, and $Pr(> F) < 2.2e - 16$. Since P-value is less

than 0.05, we can conclude that our training, testing and validation images from super 2 datasets are significantly different from each other.

CHAPTER 4
EXPERIMENTS AND RESULTS

4.1 Overview

In order to capture bee traffic, we need to be able to train and find the best performing model that would be able to accurately classify an actual "BEE" from a "NO-BEE". For this purpose, we trained and evaluated the performance of 10 Neural Networks, SVM and Random Forests(20,40,60,80,100 trees). Out of the 10 neural networks, 9 neural networks were chosen from the best performing models designed by students who participated in the Bee classification project as a part of CS6600 - Intelligent Systems curriculum. This chapter presents the performance of Deep learning models, as well as traditional machine learning models, ran on Generalized and Non-generalized datasets.

4.2 Model designs and their parameters

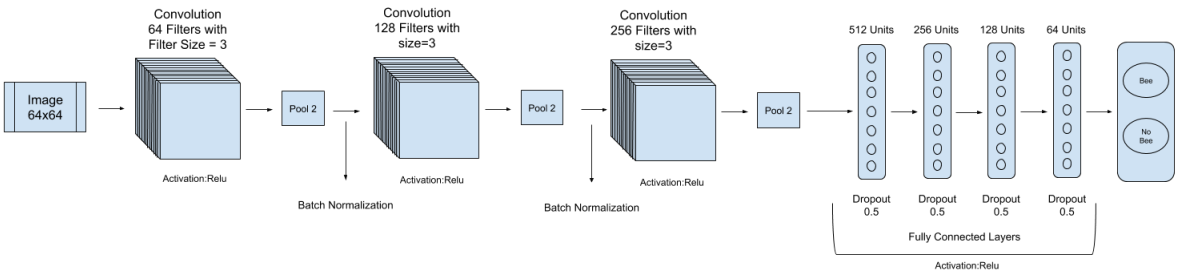

Fig. 4.1: Best performing CNN on 1-super dataset

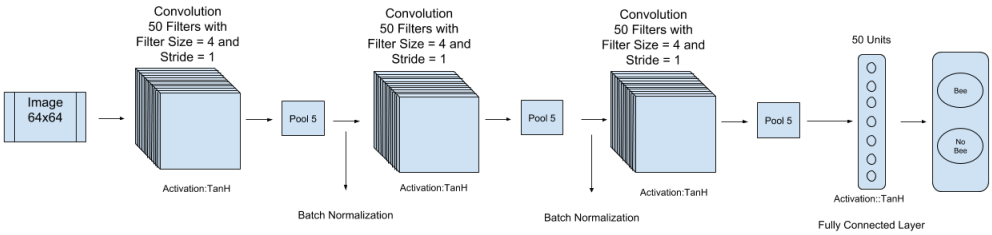

Fig. 4.2: Best performing CNN on 2-super dataset

4.2.1 Other Networks

We also used other CNN designs with different activation functions other than Relu, different filter size, number of filters, Pooling Kernel Size, number of convolution layers, and optimizers to gauge and experiment with performance tuning on our datasets. ConvNet1, ConvNet2, ConvNet3, ConvNet4, ConvNet5, ConvNet6, ConvNet7, ConvNet8, ConvNet9, and ConvNet10 are trained on BEE2 dataset and their results are shown in the below sections. The architectural specifications for these models can be found in the appendix A.

4.2.2 State of the art networks

In order to find out how our models stand against state of the art networks in terms of performance, we trained and validated 2 such networks on "Generalized" dataset on both 1-super and 2-super images. These networks are known as ResNet or Deep residual networks [22], and VGG [12]. Their performance has been evaluated in the sections further.

4.2.3 Machine Learning Models

SVMs

We trained Linear SVMs using an 'L2' penalty, a squared hinge loss function, a tolerance of 0.0001, 'ovr' as multiclass strategy and a maximum iteration of 1000 on both Generalized and Combined dataset to compare their performance with Neural networks. A 3 channel image of size 64x64 is flattened to a 1 dimension array of 12288, normalized using the min-max normalization and fed into the classifier.

Random Forests

We trained Random Forest classifiers with 20,40,60,80, and 100 trees, criterion as 'gini', a minimum number of samples required to split as 2 and a minimum number of samples required to be at a leaf node to be 2. A 3 channel image of size 64x64 is flattened to a 1 dimension array of 12288, normalized using the min-max normalization and fed into the classifier.

4.3 Results

4.3.1 Performance on Generalized Set

Table 4.1: Validation accuracy on 1-super Generalized Dataset

Model	**BEE Accuracy**	**NO-BEE Accuracy**	**Overall Accuracy**
ConvNet10	94.34%	93.24%	94.08%
ConvNet1	87.49%	96.09%	89.76%
ConvNet5	88.44%	88.44%	88.84%
VGG	78.09%	95.12%	86.60%
ConvNet7	81.64%	88.85%	83.39%
ConvNet6	72.02%	86.45%	75.53%
ConvNet2	67.99%	93.66%	74.82%
ConvNet4	66.48%	95.87%	73.44%
ResNet	38.11%	99.70%	68.90%
ConvNet9	60.49%	89.42%	67.53%
ConvNet3	0%	100%	24.31%
ConvNet8	0%	100%	24.31%
SVM	87.66%	51.06%	69.36%
RF(20)	46.62%	97.26%	71.94%
RF(40)	49.53%	97.82%	73.67%
RF(60)	50.07%	98.34%	74.205%
RF(80)	49.90%	97.86%	73.88%
RF(100)	50.27%	98.31%	74.29%

Table 4.2: Validation accuracy on 2-super Generalized Dataset

Model	**BEE Accuracy**	**NO-BEE Accuracy**	**Overall Accuracy**
ConvNet1	75.59%	81.31%	78.90%
ConvNet4	69.98%	94.84%	78.17%
ConvNet6	65.73%	84.36%	76.51%
ConvNet2	64.34%	90.51%	75.92%
ConvNet5	57.37%	90.06%	73.77%
ResNet	64.10%	83.37%	73.73%
ConvNet9	52.04%	87.49%	72.55%
ConvNet7	60.26%	84.45%	72.35%
VGG	60.78%	82.54%	71.66%
ConvNet10	51.77%	86.04%	68.90%
ConvNet3	0%	100%	57.86%
ConvNet8	0%	100%	57.86%
SVM	87.10%	41.95%	64.53%
RF(20)	35.36%	89.89%	62.62%
RF(40)	37.34%	87.83%	62.58%
RF(60)	39.05%	88.99%	64.02%
RF(80)	37.84%	88.34%	63.09%
RF(100)	37.71%	88.43%	63.07%

4.3.2 Performance on Combined Set

Table 4.3: Validation accuracy on 1-super Combined Dataset

Model	**BEE Accuracy**	**NO-BEE Accuracy**	**Overall Accuracy**
ConvNet10	91.76%	95.56	93.66%
ConvNet1	91.22%	96.55%	93.88%
ConvNet5	91.75%	94.03%	92.89%
ConvNet7	92.8%	94.76%	93.80%
ConvNet6	89.37%	95.72%	92.54%
ConvNet2	93.77%	94.35%	94.06%
ConvNet4	80.11%	96.68%	88.39%
ConvNet9	84.44%	95.85%	90.16%
ConvNet3	0%	100%	50.00%
ConvNet8	0%	100%	50.00%
SVM	67.02%	90.10%	78.56%
RF(20)	84.71%	93.65%	89.18%
RF(40)	87.07%	93.49%	90.28%
RF(60)	87.36%	94.16%	90.76%
RF(80)	87.49%	93.68%	90.58%
RF(100)	87.07%	93.93%	90.50%

Table 4.4: Validation accuracy on 2-super Combined Dataset

Model	**BEE Accuracy**	**NO-BEE Accuracy**	**Overall Accuracy**
ConvNet10	98.53%	96.85%	97.69%
ConvNet1	91.96%	97.70%	94.83%
ConvNet4	87.99%	96.06%	92.03%
ConvNet6	87.20%	94.93%	91.06%
ConvNet2	95.96%	94.73%	95.35%
ConvNet7	93.13%	95.26%	94.20%
ConvNet5	89.56%	94.93%	92.24%
ConvNet9	91.89%	95.19%	93.55%
ConvNet3	100%	0%	50.00%
ConvNet8	0%	100%	50.00%
SVM	91.93%	75.20%	83.56%
RF(20)	96.16%	95.93%	96.04%
RF(40)	96.66%	96.00%	96.33%
RF(60)	97.43%	95.83%	96.63%
RF(80)	97.16%	95.70%	96.43%
RF(100)	97.46%	95.93%	96.69%

4.4 Bee traffic Analysis

Utilizing the best models from Combined Dataset for 1-super and 2-super, the number of bees are counted in each video using algorithm shown in Fig.4.3. Out of 1000's of videos extracted from the BeePi setups, we decided to extract a subset of videos from all 4 hives on the basis of 4 time periods as described in the dataset section. Using a MOG background subtraction method [23], we detected motion between frames in a video. The coordinates of these detected motions are used to extract out smaller cropped images that contain the subject that moved during the video. These images are normalized by dividing each pixel with a maximum pixel value of 255, to make sure the training process is smooth and sent

through a ConvNet which classifies the image as a BEE or NO-BEE. Collectively using this flow, we are able to extract out the number of bees in every video. The bee motion count of these videos are grouped according to the hour of the day and persisted in a CSV file. We are able to extract the count of bee motions over a period of time for each hive, both for 1-super videos and 2-super videos.

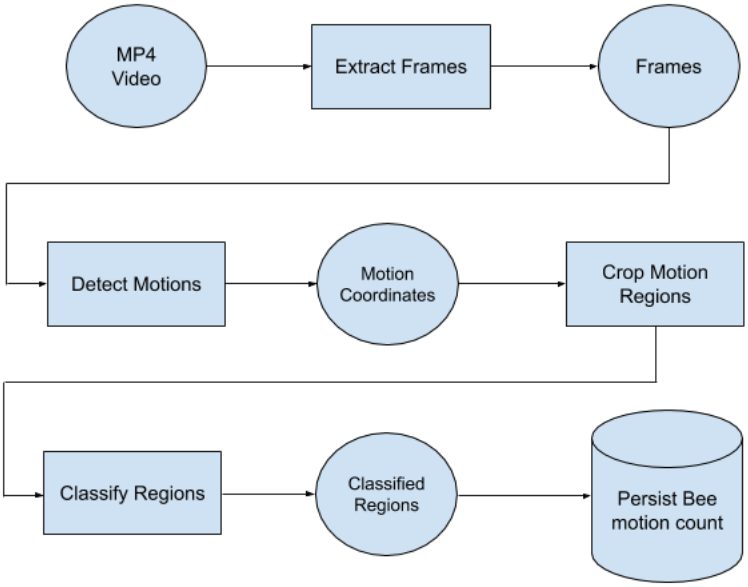

Fig. 4.3: Illustration of the algorithm used to extract bee motion count data from BeePi videos

Using the CSV data we can plot a graph of bee motion count with respect to each day during the whole beekeeping season and look for possible patterns that may help us understand hive behavior. Figure 4.5,Figure 4.6, Figure 4.7,Figure 4.8 present the bee traffic behaviour of all 4 BeePi's deployed.

After the BeePis were deployed, an inspection was conducted every 2 weeks. The inspection comprised of collecting all the data that was captured from the BeePi Systems, and manually assessing the health of the hives as a Beekeeper. The assessment of the hive was done by dismantling the hive for a short period and collecting information such has Bee Larvae, Bee Eggs, etc. A huge amount of this information was logged in a diary where observations pertaining to the encodes described in 4.5 were logged. We translated this information into quantitative information that can be compared with the bee motion count plots. The quantitative information contains egg and larvae count seen on each inspection date. We plotted these for each hive and compared them with the bee motion count plots to get some more insight.

Table 4.5: Bee inspection diary encoding

Code	**Meaning**
RS	Right Side
LS	Left Side
DC	Drawn Comb
LH	Liquid Honey
EG	Eggs
EGG	Lots of eggs
PL	Pollen
FR	Frame
LV	Larvae
LVV	Lots of larvae
CB	Capped brood
CH	Capped honey
DB	Drone brood

Using these encodings, observations were logged in a format shown in Fig 4.4

```
May 05, 2018

Richard's place; hive inspection

R4.8

BeePi working

FR 10: empty on both sides
FR 9: LS: some DC with LH; RS: empty
FR 8: LS: EG, PL; RS: some DC with LH
FR 7: LS: EG, PL, LH, DC, lots of EG; RS: EG; LH, DC
FR 6: LS: EG, LH, DC; RS: EG, LH, DB; queen cage empty
FR 5: LS: DC with LH, No EG; RS: DC wiht LH, EG, PL
FR 4: LS: empty; RS: some DC with LH
FR 3: LS: empty; RS: empty

R4.5

BeePi working

FR 10: LS: empty; RS: empty
FR 9: LS: some DC with LH; RS: empty
FR 8: LS: some DC with PL and LH, no EG; RS: some DC with PL, LH, no EG
FR 7: LS: EG, DC, LH, PL; RS: DC, LH, EG, PL;
FR 6: LS: EG, DC, PL, LH; RS: EG, DC, PL, LH; queen cage empty
FR 5: LS: some DC, LH; RS: some DC, LH, PL, LH
FR 4: LS: empty; RS: some DC
FR 3: LS: empty; RS: empty

R4.10

BeePi working

FR 10: LS: empty; RS: empty
FR 9: LS: DC, LH, some PL; RS: some DC;
FR 8: LS: EG, PL, LH, DC; RS: DC, PL, LH, some EG
FR 7: LS: DC, LV, some LH, EG; RS: EG, some PL, some LH;
FR 6: LS: some DC with LH; RS: DC, some PL, some LH; queen cage empty
FR 5: LS: empty; RS: some DC with LH;
FR 4: LS: empty; RS: empty
FR 3: LS: empty; RS: empty
```

Fig. 4.4: Bee inspection log excerpt from the diary

To co-relate the observations logged in the diary, we picked out two important components that contribute to bee population - eggs and larvae. The encodes EG,EGG,LV and LVV indicate observations of eggs and larvae on the either side of a frame. This encodes were then quantified in a way, we could create a small dataset with eggs and larvae count associated to a time. We decided to give a score of 1 to EG/LV, and 3 to EGG/LVV. For each inspection we could now associate each frame of a hive with egg count and larvae

count. Since, a super can have multiple frames, we decided to aggregate the egg and larvae count to hive level. Having done that, that gave us eggs and larvae count for each hive for every inspection. We could arrange this data and plot it against each inspection date, to analyze eggs and larvae count activity over a period of time as shown in Fig 4.5, 4.6, 4.7, and 4.8

Behaviour of hives can be understood through these graphs by visualizing graphs using egg/larvae data or Bee traffic data of a healthy hive. When a BeePi is deployed, the bee population starts small, the queen bee lays eggs, and it is expected for the bee population to grow. This will directly be indicated by the count of bee motions captured in our data. So, an upward movement of bee motion count in a graph would indicate growth of the bee colony in that hive. It can be observed from the graphs that hives 4.5 and 4.10, continued to show erratic behaviour, with bee population not growing consistently. Hives 4.7 and 4.8 showed consistent growth till the end of the period.

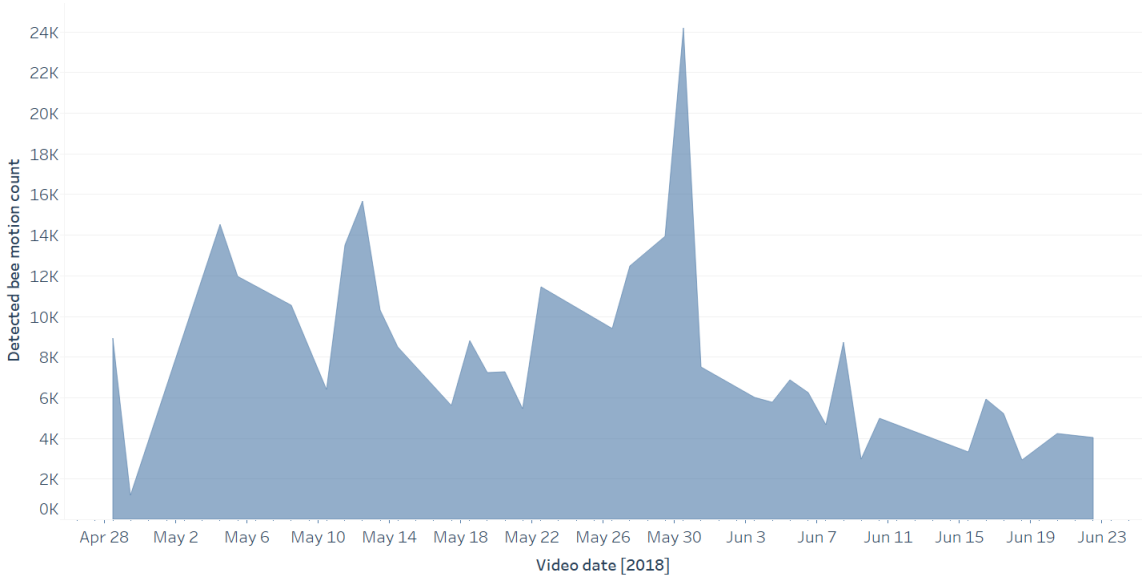

Fig. 4.5: Hive 4.5 - Count of bee motions detected with respect to date

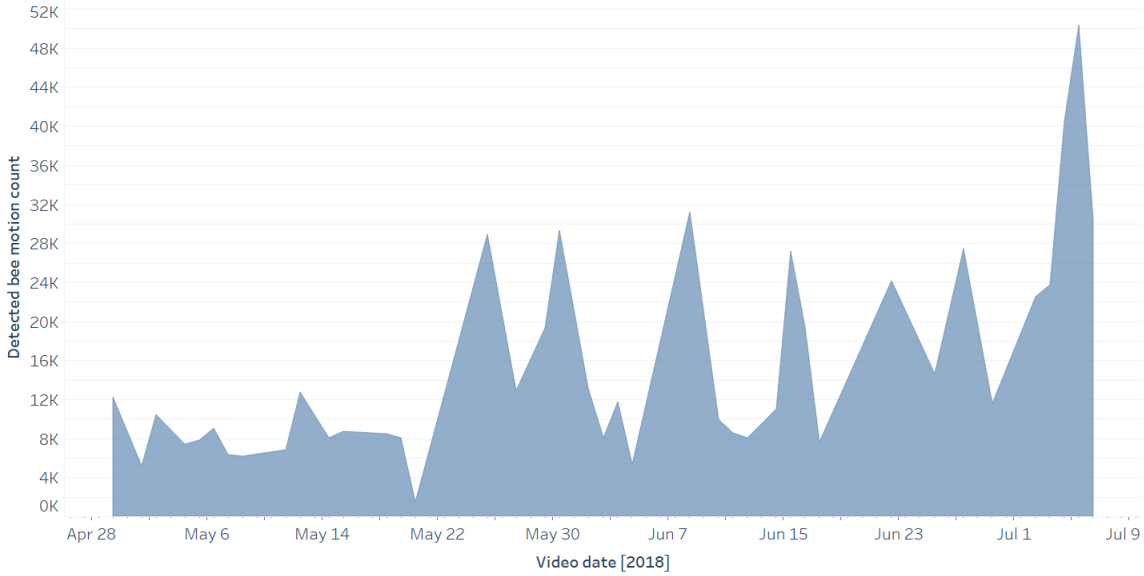

Fig. 4.6: Hive 4.7 - Count of bee motions detected with respect to date

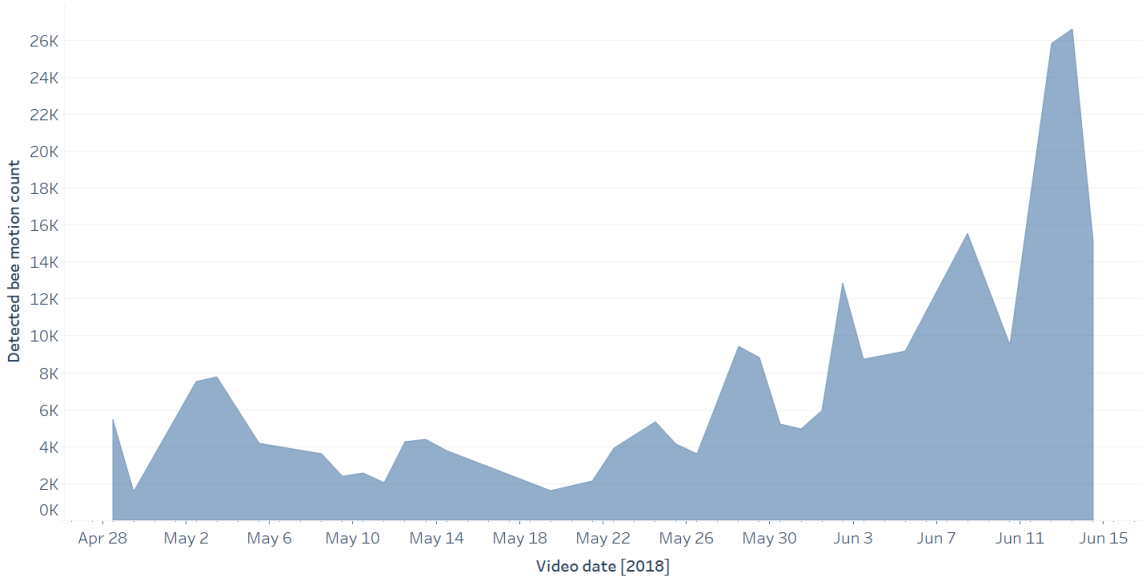

Fig. 4.7: Hive 4.8 - Count of bee motions detected with respect to date

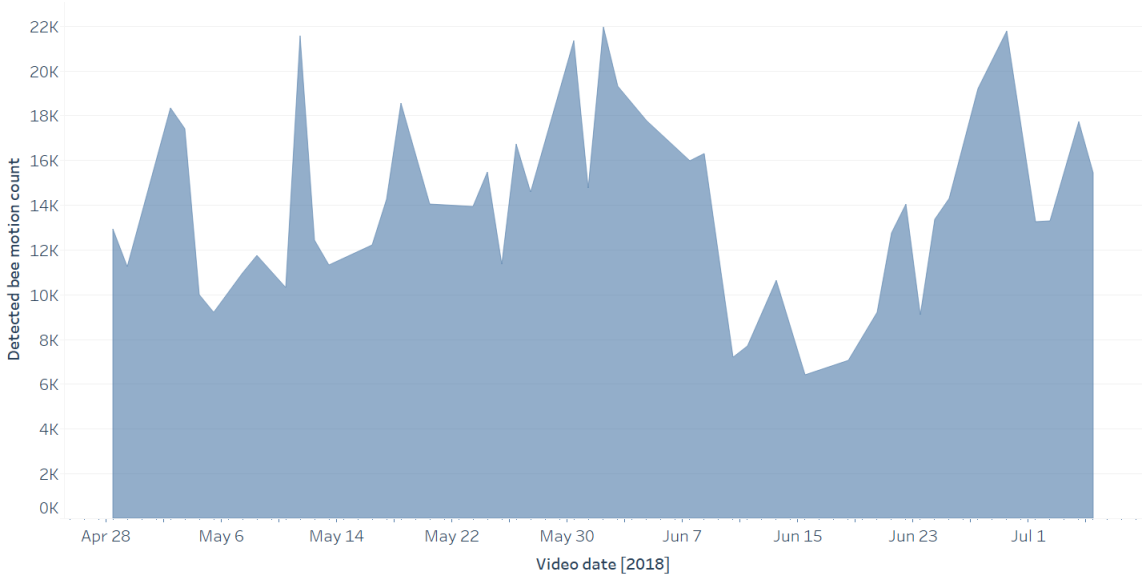

Fig. 4.8: Hive 4.10 - Count of bee motions detected with respect to date

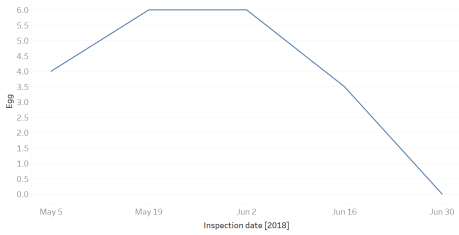

Fig. 4.9: Egg count - Hive 4.5

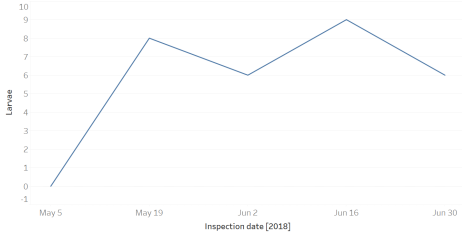

Fig. 4.10: Larvae count - Hive 4.5

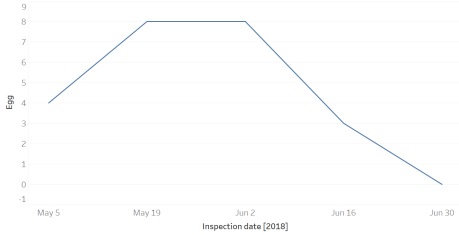

Fig. 4.11: Egg count - Hive 4.7

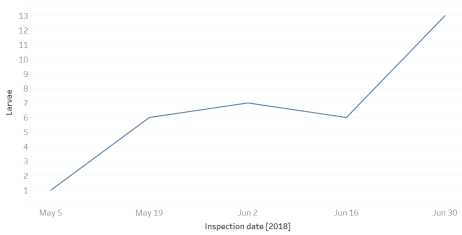

Fig. 4.12: Larvae count - Hive 4.7

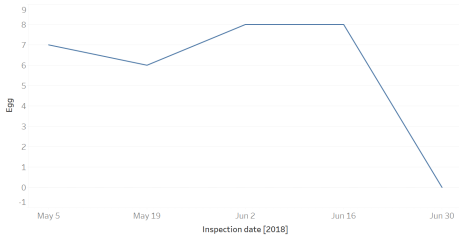

Fig. 4.13: Egg count - Hive 4.8

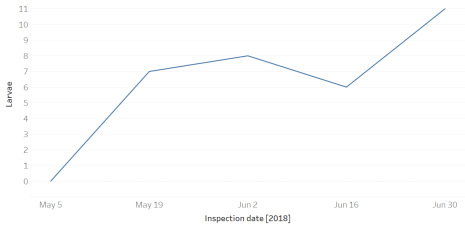

Fig. 4.14: Larvae count - Hive 4.8

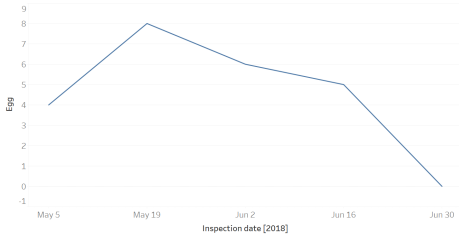

Fig. 4.15: Egg count - Hive 4.10

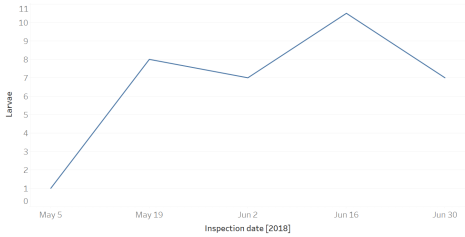

Fig. 4.16: Larvae count - Hive 4.10

It can be observed that there is a co-relation of egg and larvae count in the figures above. The relationship between egg and larvae count appears to be inversely proportional, and it makes sense because as eggs increase and then hatch, the egg count decreases and larvae count increases. Thus, an important observation to note is that a consistent increase in the egg count along with a consistent increase in the larvae count indicates a possibly healthy growing hive. In the figures above, we can see that as the egg count decreases and the larvae count increases in hives 4.7 and 4.8, whereas, they both egg count and larvae count decreases in hives 4.5 and 4.10.

Going back to our observation and analysis of hive health from the bee motion count plots, we can co-relate manual hive inspections with our automated analysis. The bee traffic movement is upward in 4.7 and 4.8, indicating a thriving bee-colony.

CHAPTER 5

CONCLUSION AND FUTURE WORK

For bee-hive health monitoring, this thesis explores a collection of methods that can be used to analyze bee traffic, by taking advantage of Deep neural networks that were trained during the process. We explored other traditional classification methods as well and compared them with ConvNets. It can be observed that traditional machine learning methods like Random Forests perform as well as the ConvNets on the "Combined" dataset, however, they fail to match ConvNets performance on the "Generalized" dataset. This observation can be construed to conclude that ConvNets tend to generalize better than traditional machine learning methodologies.

In order to recognize bees accurately, it is paramount for the data to be consistent and clean. Even though deep learning helps us get a more generalized classifier for images, it is as good as the data it gets and this can be observed from the dip in performance of the ConvNets, SVMs, and Random Forests on the "Generalized" set. Since Video data collected from Hives 4.8 and 4.10 comprised of a different background, and hive 4.10's videos being blurry, the models' performance decreases, as they were trained on negative data that is completely different from the training data.

Using the models, we were able to gather bee traffic data across a span of months and connect this data with manual inspections. We can deploy these models in BeePi's and track the bee motion counts live by connecting them to a remote server, where we can upload the count of bee motions observed for a given time, and collectively construct a live graph that can be observed from a remote location.

To validate and train a model that can use bee traffic data to assess a hive's health, we need many more manual inspections and BeePis. In our future work, we propose deployment of beehive monitoring code and collecting more data, along with increasing manual inspections. More manual inspections will result in more data being collected which can

then be used to analyze a hive's behavior. We also need more BeePi setups, preferably in a different setting, to get more variation in our data.

APPENDIX

A.1 Architecture of ConvNet Models used in the experiments

Table A.1: Detailed specification of ConvNet1

Configuration of ConvNet1		
Layers		**Specification**
Layer 1	Conv-2D	filters = 50, filterSize = 4, strides = 1, activation = `tanh`, bias = `True`, biasInit = `zeros`, weightsInit=uniform scaling, regularizer = `none`, weightDecay = 0.001
Layer 2	Maxpool-2D	kernelSize = 5, strides = 1
	Conv-2D	filters = 50, filterSize = 4, strides = 1, activation = `tanh`, bias = `True`, biasInit = `zeros`, weightsInit=uniform scaling, regularizer=`none`,weightDecay=0.001
Layer 3	Maxpool-2D	kernelSize = 5, strides = 1
	Conv-2D	filters = 50, filterSize = 4, strides = 1, activation = `tanh`, bias = `True`, biasInit = `zeros`, weightsInit = uniform scaling, regularizer = `none`,weightDecay=0.001
Layer 4	Maxpool-2D	kernelSize = 5, strides = 1
	FC	number of units = 50, activation = `tanh`
Layer 5	FC	number of units = 2, activation = `softmax`
Regression		Optimizer = `SGD`, Learning Rate = 0.01, Loss = `Categorical Crossentropy`

Table A.2: Detailed specification of ConvNet2

Configuration of ConvNet2		
Layers		**Specification**
Layer 1	Conv-2D	filters = 32, filterSize = 5, strides = 1, activation = `relu`, bias = `True`,biasInit = `zeros`, weightsInit=uniform scaling, regularizer = `none`, weightDecay = 0.001
Layer 2	Batch Normalization	
	Maxpool-2D	kernelSize = 4, strides = `none`
	FC	number of units = 100, activation = `relu`
Layer 3	Dropout	keep probability = 0.5
	FC	number of units = 2, activation = `softmax`
Regression		Optimizer = `SGD`, Learning Rate=0.01,Loss=`Categorical Crossentropy`

Table A.3: Detailed specification of ConvNet3

Configuration of ConvNet3		
Layers		**Specification**
Layer 1	Conv-2D	filters = 32, filterSize = 3, strides = 1, activation =`relu`,bias=`True`,biasInit=`zeros`, weightsInit=uniform scaling, regularizer=`none`,weightDecay=0.001
Layer 2	Maxpool-2D	kernelSize = 2, strides = 1
	Conv-2D	filters = 64, filterSize = 3, strides = 1, activation = `relu`, bias = `True`, biasInit = `zeros`, weightsInit = uniform scaling, regularizer = `none`, weightDecay = 0.001
Layer 3	Conv-2D	filters = 64, filterSize = 4, strides = 1, activation = `relu`, bias = `True`, biasInit = `zeros`, weightsInit = uniform scaling, regularizer = `none`, weightDecay = 0.001
Layer 4	Maxpool-2D	kernelSize = 2, strides = 1
	FC	number of units = 512, activation = `relu`
Layer 5	Dropout	keep probability = 0.5
	FC	number of units = 2, activation = `softmax`
Regression		Optimizer = `Adam`, Learning Rate=0.01,Loss=`Categorical Crossentropy`

Table A.4: Detailed specification of ConvNet4

Configuration of ConvNet4		
Layers		**Specification**
Layer 1	Conv-2D	filters = 20, filterSize = 5, strides = 1, activation =`relu`,bias=`True`,biasInit=`zeros`, weightsInit=uniform scaling, regularizer=`none`,weightDecay=0.001
Layer 1	Conv-2D	filters = 25, filterSize = 5, strides = 1, activation =`relu`,bias=`True`,biasInit=`zeros`, weightsInit=uniform scaling, regularizer=`none`,weightDecay=0.001
Layer 3	Maxpool-2D	kernelSize = 2, strides = 1
	FC	number of units = 100, activation = `sigmoid`
Layer 4	FC	number of units = 2, activation = `softmax`
Regression		Optimizer = `SGD`, Learning Rate=0.01,Loss=`Categorical Crossentropy`

Table A.5: Detailed specification of ConvNet5

Configuration of ConvNet5		
Layers		**Specification**
Layer 1	Conv-2D	filters = 10, filterSize = 10, strides = 1, activation =`tanh`,bias=`True`,biasInit=`zeros`, weightsInit=uniform scaling, regularizer=`none`,weightDecay=0.001
Layer 2	Maxpool-2D	kernelSize = 4, strides = 1
	Conv-2D	filters = 30, filterSize = 3, strides = 1, activation =`tanh`,bias=`True`,biasInit=`zeros`, weightsInit=uniform scaling, regularizer=`none`,weightDecay=0.001
Layer 4	Maxpool-2D	kernelSize = 4, strides = 1
	FC	number of units = 100, activation = `sigmoid`
Layer 5	FC	number of units = 2, activation = `softmax`
Regression		Optimizer = `SGD`, Learning Rate=0.01,Loss=`Categorical Crossentropy`

Table A.6: Detailed specification of ConvNet6

Configuration of ConvNet6		
Layers		**Specification**
Layer 1	Conv-2D	filters = 20, filterSize = 7, strides = 1, activation =`relu`,bias=`True`,biasInit=`zeros`, weightsInit=uniform scaling, regularizer=`none`,weightDecay=0.001
Layer 2	Maxpool-2D	kernelSize = 3, strides = 1
	Conv-2D	filters = 20, filterSize = 5, strides = 1, activation =`relu`,bias=`True`,biasInit=`zeros`, weightsInit=uniform scaling, regularizer=`none`,weightDecay=0.001
Layer 4	Maxpool-2D	kernelSize = 2, strides = 1
	FC	number of units = 100, activation = `sigmoid`
Layer 5	FC	number of units = 2, activationModel7 = `sigmoid`
Regression		Optimizer = `SGD`, Learning Rate=0.01,Loss=`Categorical Crossentropy`

Table A.7: Detailed specification of ConvNet7

Configuration of ConvNet7		
Layers		**Specification**
Layer 1	Conv-2D	filters = 20, filterSize = 5, strides = 1, activation = `relu`, bias = `True`, biasInit = `zeros`, weightsInit = uniform scaling, regularizer = `none`, weightDecay = 0.001
Layer 2	Maxpool-2D	kernelSize = 2, strides = 1
	Conv-2D	filters = 15, filterSize = 5, strides = 1, activation = `relu`, bias = `True`, biasInit = `zeros`, weightsInit = uniform scaling, regularizer = `none`, weightDecay = 0.001
Layer 4	Maxpool-2D	kernelSize = 2, strides = 1
	FC	number of units = 15, activation = `relu`
Layer 5	FC	number of units = 2, activation = `sigmoid`
Regression		Optimizer = `SGD`, Learning Rate=0.01,Loss=`Categorical Crossentropy`

Table A.8: Detailed specification of ConvNet8

Configuration of ConvNet8		
Layers		**Specification**
Layer 1	Conv-2D	filters = 20, filterSize = 5, strides = 1, activation =`relu`,bias=`True`,biasInit=`zeros`, weightsInit=uniform scaling, regularizer=`none`,weightDecay=0.001
Layer 2	Maxpool-2D	kernelSize = 2, strides = 1
	Conv-2D	filters = 50, filterSize = 5, strides = 1, activation =`relu`,bias=`True`,biasInit=`zeros`, weightsInit=uniform scaling, regularizer=`none`,weightDecay=0.001
Layer 3	Maxpool-2D	kernelSize = 2, strides = 1
	Conv-2D	filters = 30, filterSize = 5, strides = 1, activation =`relu`,bias=`True`,biasInit=`zeros`, weightsInit=uniform scaling, regularizer=`none`,weightDecay=0.001
Layer 4	Maxpool-2D	kernelSize = 2, strides = 1
	FC	number of units = 140, activation = `relu`
Layer 5	FC	number of units = 80, activation = `sigmoid`
Layer 6	FC	number of units = 2, activation = `sigmoid`
Regression		Optimizer = `SGD`, Learning Rate=0.01,Loss=`Categorical Crossentropy`

Table A.9: Detailed specification of ConvNet9

Configuration of ConvNet9		
Layers		**Specification**
Layer 1	Conv-2D	filters = 32, filterSize = 3, strides = 1, activation =`relu`,bias=`True`,biasInit=`zeros`, weightsInit=uniform scaling, regularizer=`none`,weightDecay=0.001
Layer 2	Conv-2D	filters = 64, filterSize = 3, strides = 1, activation =`relu`,bias=`True`,biasInit=`zeros`, weightsInit=uniform scaling, regularizer=`none`,weightDecay=0.001
Layer 3	Maxpool-2D	kernelSize = 2, strides = 1
	FC	number of units = 256, activation = `sigmoid`
Layer 4	FC	number of units = 2, activation = `sigmoid`
Regression		Optimizer = `SGD`, Learning Rate=0.01,Loss=`Categorical Crossentropy`

Table A.10: Detailed specification of ConvNet10

Configuration of ConvNet10		
Layers		**Specification**
Layer 1	Conv-2D	filters = 64, filterSize = 3, strides = 1, activation =`relu`,bias=`True`,biasInit=`zeros`, weightsInit=uniform scaling, regularizer=`none`,weightDecay=0.001
Layer 2	Maxpool-2D	kernelSize = 3, strides = 1
	Conv-2D	filters = 128, filterSize = 3, strides = 1, activation =`relu`,bias=`True`,biasInit=`zeros`, weightsInit=uniform scaling, regularizer=`none`,weightDecay=0.001
Layer 3	Maxpool-2D	kernelSize = 3, strides = 1
	Conv-2D	filters = 256, filterSize = 3, strides = 1, activation =`relu`,bias=`True`,biasInit=`zeros`, weightsInit=uniform scaling, regularizer=`none`,weightDecay=0.001
Layer 4	Maxpool-2D	kernelSize = 2, strides = 1
	Dropout	keep probability = 0.5
	FC	number of units = 512, activation = `relu`
Layer 5	Dropout	keep probability = 0.5
	FC	number of units = 256, activation = `relu`
Layer 6	Dropout	keep probability = 0.5
	FC	number of units = 128, activation = `relu`
Layer 7	Dropout	keep probability = 0.5
	FC	number of units = 64, activation = `relu`
Layer 8	Dropout	keep probability = 0.5
	FC	number of units = 2, activation = `relu`
Regression		Optimizer = `adam`, Learning Rate=0.0001,Loss=`Categorical Crossentropy`

REFERENCES

[1] M. Reno, R. Broderick, and L. Blakely, "Machine learning for rapid qsts simulations using neural networks," 06 2017.

[2] K. Gopalakrishnan, "Deep learning in pavement image analysis and automated distress detection: A review," *Data*, vol. 3, no. 3, 2018.

[3] USDA. Usda releases results of new survey on honey bee colony health. [Online]. Available: https://www.usda.gov/media/press-releases/2016/05/12/usda-releases-results-new-survey-honey-bee-colony-health

[4] S. G. Potts, S. P. M. Roberts and R. Dean, G. Marris and M. A. Brown and R. Jones and Peter Neumann and Josef Settele, "Declines of managed honey bees and beekeepers in europe," *Journal of Apicultural Research*, vol. 49, pp. 15–22, 2010.

[5] A. R. McLellan, "Honeybee colony weight as an index of honey production and nectar flow: A critical evaluation," *Journal of Applied Ecology*, vol. 14, no. 2, pp. 401–408, 1977. [Online]. Available: http://www.jstor.org/stable/2402553

[6] M. Pham-Delègue, A. Decourtye, L. Kaiser, and J. Devillers, "Behavioural methods to assess the effects of pesticides on honey bees," *Apidologie*, vol. 33, no. 5, pp. 425–432, 2002.

[7] V. A. Kulyukin and S. K. Reka, "Toward sustainable electronic beehive monitoring: Algorithms for omnidirectional bee counting from images and harmonic analysis of buzzing signals," *Engineering Letters*, vol. 24, p. 3, 2016.

[8] V. A. Kulyukin, "In situ omnidirectional vision-based bee counting using 1d haar wavelet," *International MultiConference of Engineers and Computer Scientists*, vol. 1, Mar. 2017.

[9] A. Tiwari, "A deep learning approach to recognizing bees in video analysis of bee traffic," Master's thesis, Utah State University, Logan, UT, 2018.

[10] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 818–833.

[11] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: An astounding baseline for recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2014.

[12] K. Simonyan, A. Zisserman, "Very deep convolutional networks for large-scale image recognition." arXiv.org, 2015. [Online]. Available: https://arxiv.org/pdf/1409.1556.pdf

[13] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[14] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 2553–2561. [Online]. Available: http://papers.nips.cc/paper/5207-deep-neural-networks-for-object-detection.pdf

[15] J. Paul Cohen, G. Boucher, C. A. Glastonbury, H. Z. Lo, and Y. Bengio, "Count-ception: Counting by fully convolutional redundant counting," in *The IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct 2017.

[16] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[17] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep 1995. [Online]. Available: https://doi.org/10.1007/BF00994018

[18] Herman Van de Kerkof. Bee hive. [Online]. Available: https://patentimages.storage.googleapis.com/e1/2d/b1/ad2cb835ee8a8e/US4135265.pdf

[19] R. A. Fisher, *Statistical Methods for Research Workers*. New York, NY: Springer New York, 1992, pp. 66–70. [Online]. Available: https://doi.org/10.1007/978-1-4612-4380-9_6

[20] R. T. Warne, "A primer on multivariate analysis of variance (manova) for behavioral scientists," *Practical Assessment, Research Evaluation*, vol. 19, no. 17, 2014.

[21] M. Hall-Beyer. (2017) Glcm texture: A tutorial. [Online]. Available: https://prism.ucalgary.ca/handle/1880/51900

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[23] P. KaewTraKulPong and R. Bowden, *An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection*. Boston, MA: Springer US, 2002, pp. 135–144. [Online]. Available: https://doi.org/10.1007/978-1-4615-0913-4_11