# BEE SHADOW RECOGNITION IN VIDEO ANALYSIS OF OMNIDIRECTIONAL BEE TRAFFIC

by

Laasya Alavala

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

of

MASTER OF SCIENCE

in

Computer Science

Approved:

| | |
|---|---|
| Vladimir Kulyukin, Ph.D.<br>Major Professor | Xiaojun Qi, Ph.D.<br>Committee Member |
| Curtis Dyreson, Ph.D.<br>Committee Member | Richard S. Inouye, Ph.D.<br>Vice Provost for Graduate Studies |

UTAH STATE UNIVERSITY
Logan, Utah

2019

Copyright © Laasya Alavala 2019

All Rights Reserved

## ABSTRACT

Bee Shadow Recognition in Video Analysis of Omnidirectional Bee Traffic

by

Laasya Alavala, Master of Science

Utah State University, 2019

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

Shadows of bees are recognized as bees in video analysis of forager traffic. Thus shadow detection and elimination is proposed as necessary and important step in the process of recognizing bees. False detection of bee shadows as bees may lead to incorrect analysis of bee colony behaviour and health. This thesis proposes various computer vision techniques and Convolution Neural Network(ConvNets) to detect bees accurately. The best model is then integrated with a motion detection algorithm that works on a Raspberry Pi 3 computer which is part of BeePi. We designed and tested the bee detection accuracy with Machine Learning techniques, ConvNets and several analytical approaches. We compared our models with some of the standard state of the art models. The main objective is to find the best classifier to detect bees with less computational time, which improves the estimation of bee traffic levels.

(76 pages)

## PUBLIC ABSTRACT

Bee Shadow Recognition in Video Analysis of Omnidirectional Bee Traffic

Laasya Alavala

Over a decade ago, beekeepers noticed that the bees were dying or disappearing without any prior health disorder. Colony Collapse Disorder(CCD) has been a major threat to bee colonies around the world which affects vital human crop pollination. Possible instigators of CCD include viral and fungal diseases, decreased genetic diversity, pesticides and a variety of other factors. The interaction among any of these potential facets may be resulting in immunity loss for honey bees and the increased likelihood of collapse. It is essential to rescue honey bees and improve the health of bee colony.

Monitoring the traffic of bees helps to track the status of hive remotely. An Electronic beehive monitoring system extracts video, audio and temperature data without causing any interruption to the bee hives. This data could provide vital information on colony behaviour and health. This research uses Artificial Intelligence and Computer Vision methodologies to develop and analyze technologies to monitor omnidirectional bee traffic of hives without disrupting the colony. Bee traffic means the number of bees moving in a given area in front of the hive over a given period of time. Forager traffic is the number of bees coming in and/or leaving the hive over a time. Forager traffic is a significant component in monitoring food availability and demand, colony age structure, impacts of pests and diseases, etc on hives. The goal of this research is to estimate and keep track of bee traffic by eliminating unnecessary information from video samples.

To all the little people....

## ACKNOWLEDGMENTS

Foremost, I would like to express my sincere gratitude to my advisor Dr. Vladimir Kulyukin for the useful comments, motivation and encouragement through the learning process of this thesis. He always steered me in the right direction whenever I needed. I could not have imagined having a better advisor and mentor for my thesis.

I would like to thank the rest of my thesis committee: Dr. Xiaojun Qi and Dr. Curtis Dyreson for their continuous support and insightful comments.

Finally, I wish to thank my parents and friends for their support and continuous encouragement in hard times.

Laasya Alavala

CONTENTS

LIST OF TABLES

LIST OF FIGURES

## ACRONYMS

| | |
|---|---|
| CCD | Colony Collapse Disorder |
| EBM | Electronic Beehive Monitoring |
| MANOVA | Multivariate Analysis Of Variance |
| GLCM | Gray Level Co-occurrence Matrix |
| PCA | Principal Component Analysis |
| SFS | Sequential Feature Selection |
| UFS | Univariate Feature Selection |
| DL | Deep Learning |
| ML | Machine Learning |
| MLP | Multi Layer Perceptron |
| ConvNet | Convolutional Neural Network |
| POOL | Pooling layer |
| SGD | Stochastic Gradient Descent |
| CONV | Convolutional layer |
| FC | Fully Connected layer |
| DNN | Deep Neural Networks |

# CHAPTER 1
# INTRODUCTION

## 1.1 Background

The world's premier managed pollinator species are the Western honey bees(Apis mellifera L.). In the United States, approximately $15 billion of crops are pollinated by honey bees every year [4]. Fruits, nut and vegetable farmers raised the demand for honey bee services. The most important pollinators for agricultural crops are honey bees.

In both the USA and Europe, Apiculture has been diminished over the decades. Pathogens and other factors contribute to high bee colony losses [5, 6]. Therefore, it is vital to pursuit beekeeping rather than taking it as a laborious profession to endorse pollination and local apiculture. Abrupt loss of honey colonies has happened. Colony Collapse Disorder(CCD) has gained great attention from the public. European scientists are also working hard to explain the vast colony losses. Due to the break down of honey bee supply, the prices for fruits,nuts and vegetables have risen.

All these factors lead scientists and farmers to consider bee health as a critical issue. Many scientists and farmers are working around the world for about a decade to address this problem. Plenty of work has been done earlier for data collection and interpretation. Khoury, Barron and Myerscough [7] proposed a model in 2013 on designing a model to predict complex interactions between bees death rate and food availability. Forager traffic is a significant variable for tracking food availability and demand, impact of pesticides used, colony age structure, etc. in bee hives [8]. To analyze the impact of pesticides on bee colony health, forager activity is an important variable. Monitoring forager 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 [9]. 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 the number of bees coming in or going out of the hive over a given period of time. Human assistance, while likely to be accurate, clearly limits because it is a time-consuming process to monitor the hives continuously. Generally, most of the hives are not located in the immediate vicinity of beekeeper who tend them, so every inspection incurs a transportation cost to the bee keepers. Also, the number of invasive hive inspections often disrupt the life cycles of bee colonies.

Thus, Electronic Beehive Monitoring (EBM) systems makes it possible to automate the process of gathering abundant information about the bees behaviour without disturbing the bee colonies and increasing physical demand on beekeepers [10]. EBM systems collect video, audio and temperature data which helps to define the state of a honeybee colony. Each kind of data reveals some information about the health of hives. Analyzing this data continuously over a period of time helps to differentiate the abnormal behaviour of honey bees.

## 1.2 Related Work

The importance of monitoring bee hives and estimating its health demands research and advancement over a span for several years. EBM system helps in extracting critical information on colony behaviour without interfering any checkup on bee hives and transit expenses. EBMs robotize the process of collecting information form bee hives. Incorporating novel software solutions and improving EBMs design helps for better utilization.

Some algorithms were developed based on computer vision to solve the bee counting problem which further contributes solutions to estimate forager traffic. Two algorithms are presented for omnidirectional bee counting from images to estimate bee traffic levels. The first algorithm is based on contour detection. A list of contours and connected components of pixels are measured from an image which is binarized. The number of the contours detected indicate the count of bees on the pad. The second algorithm is based on color segmentation. Each pixel in the image is inspected for green color. If the pixel color exceeds a threshold then it is marked as a pad pixel, otherwise as a bee pixel [11]. The number of bees on the landing pad is estimated by dividing the number of bee detected pixels by a hand crafted threshold. A more accurate was designed based on the concept of 1D Haar Wavelet Spike and tested in situ on Langstroth landing pad [12].

As Deep Learning(DL) has gained momentum in several fields, with computer vision being one of the most prominent cases. Images of size 32x32 were extracted and labelled from the videos collected by BeePis. This work investigates several DL methodologies i.e. analyzed and compared the performance of MLPs and ConvNets and gave better performance results than previous approaches [13]. The drawback of this classification is shadow bee images are also classified as bees.

Detection of shadows is a challenging task as shadows have same magnitude and movement pattern similar to foreground objects. Shadows may be falsely detected as objects in object detection algorithms. Shadow detection is an important pre-processing step im many computer vision applications. Murali et. al. [14] proposed an algorithm to detect shadow pixels from the image. The image is converted to LAB color space. Since the shadow region is darker and less illuminated than other regions. So, applying threshold on separate channels gives makes a clear difference between shadow and non-shadow pixels. Shadows may also be misinterpreted in the image segmentation process. An efficient approach for shadow detection and removal was proposed by Pal et. al. [15]. Their algorithm is based on HSV color model and OTSU thresholding.

Extracting ideal features that can reflect the intrinsic content of the images is a challenging problem in computer vision. Feature extraction is related to dimensionality reduction. Some of its applications are latent semantic analysis, data compression, decomposition and projection, and pattern recognition, etc. The main objective of feature extraction is to have the most relevant information represented in a lower dimensionality space. It can also enhance the speed and effectiveness of supervised learning. Mohanaiah et. al. [16] presented an application of gray level co-occurrence matrix to extract second order statistical textural features for images.

Image classification is a challenging task due to the quality distortions in images such as noise, lighting, blur, scaling, etc. Human image recognition is typically rapid and more accurate when compared to machines,as the internal representation of images are largely independent of viewpoint and orientation. Deep Neural Networks(DNNs) have gained excellent performance over a decade. DNNs have the ability to describe behaviour on a more fine-grained level. They are more complex and use multiple layers to progressively extract high level features to represent more abstract patterns in the images which are unrecognizable to humans. Deep Neural Networks(DNNs) create images of high perceptual quality and a neural representation to divide and combine the features of arbitrary images. Deep learning(DL) have improved the state-of-the-art techniques in speech recognition, object recognition, visual object detection and in many other domains.

Convolution Neural Networks(ConvNets) are part of DNNs, have appeared to outperform all other techniques in image classification. Zeiler and Fergus [17] introduced a novel visualization technique that gives insight into the function of intermediate feature layers and operations of the classifier.

Krizhevsky, Sutskever and Hinton [18] presented their findings on applying deep neural networks to classify large image datasets. They trained a deep neural network to classify 1.2 million images in the ImageNet LSVRC-2010 contest into 1000 different classes. The neural network has million parameters and 650,000 neurons, consists of five convolution layers, some of which are followed by max-pooling layers, and three fully connected layers with a final 1000-way softmax. This network showed to have best results on other datasets such as MNIST, CIFAR, etc. This architecture is called as AlexNet.

Deep neural networks are difficult to train as they are learning machines with lots of parameters, and the building blocks are non-linear units. As they are stacked in layers, the objective function becomes highly non-complex, so optimization is hard. He et. al [19] contributed a novel residual learning framework to ease the training of deeper networks. The researchers explicitly reformulated the layers as learning residual functions with reference to the layer inputs. An ensemble of these residual nets achieves 3.57% error rate on ImageNet test set. Their work proclaimed to have better generalization over other popular networks.

Szegedy et. al. [2] proposed a deep convolution neural network, called Inception. The main trademark is to improve the utilization of computational resources inside the network. Inception is incarnated in GoogleLeNet, a 22 layers deep network. The major benefit of this method is a significant quality gain at modest increase of computational requirements compared to shallower and less wide networks.

The success of region proposed methods and region-based ConvNets(R-CNNs) has improved the growth in object detection. One of such work introduces a Region Proposal Network(RPN), which shares full-image convolutional features with detection network. It simultaneously predicts object bounds and objectness scores at each position. Fast R-CNNs use the trained RPNs, which generate high quality-region proposals for detection [20].

## 1.3 Current Work

In this thesis, we extended the previous work by adding a new category “SHADOW BEE” in classifying the bees. The image size is also increased from 32x32 to 64x64 to extract more information from images for better bee detection. This thesis has been organized as follows. Chapter 2 explains about the process of collecting and processing the data, different datasets we used and analyzed by MANOVA. Chapter 3 describes the algorithms for detecting the shadows in images. Chapter 4 illustrates the textural features of images and feature selection techniques. Chapter 5 discusses ConvNets, State of the art models and Machine Learning (ML) techniques in detail. Chapter 6 goes in detail about the experiments and their results. Chapter 7 concludes the results from the experiments performed and potential future work of this thesis.

# CHAPTER 2
# MATERIALS AND DATASETS

## 2.1 Overview

This section provides a detailed description of the hardware components used for data collection, how the datasets are obtained on which the classifiers are trained to detect the bees in videos and MANOVA analysis on data.

## 2.2 BeePi and Data Collection

Each BeePi consists of a Raspberry Pi computer, a temperature sensor, a miniature camera, an equipment clock, a breadboard, a battery, solar panel wires and controllers. The principal goal of this BeePi design is reproducibility. This equipment is placed on top of all the hive frames. The hardware components of BeePi are shown in 2.1. Each BeePi EBM system is equipped to collect video, temperature and 27.0 GB of audio data in distinct climatic conditions. Likewise, four BeePi EBM systems were designed and deployed in Northern Utah apiaries. All the hardware components in BeePi are intended to design for Langsthroth hives [21].

The Raspberry Pi 3 computer which is used for integration with BeePi has 1GB of RAM and 4 cores provided for handling recordings with the classifier. The data used for training the testing Neural Network models and other Machine Learning techniques is obtained from the recordings stocked by the solar-powered, a multi-sensor EBM system, called BeePi.

Four BeePi's(henceforth referred to as 4.5,4.7,4.8 and 4.10) were set up in summer of 2018 for four bee hives to collect the data.

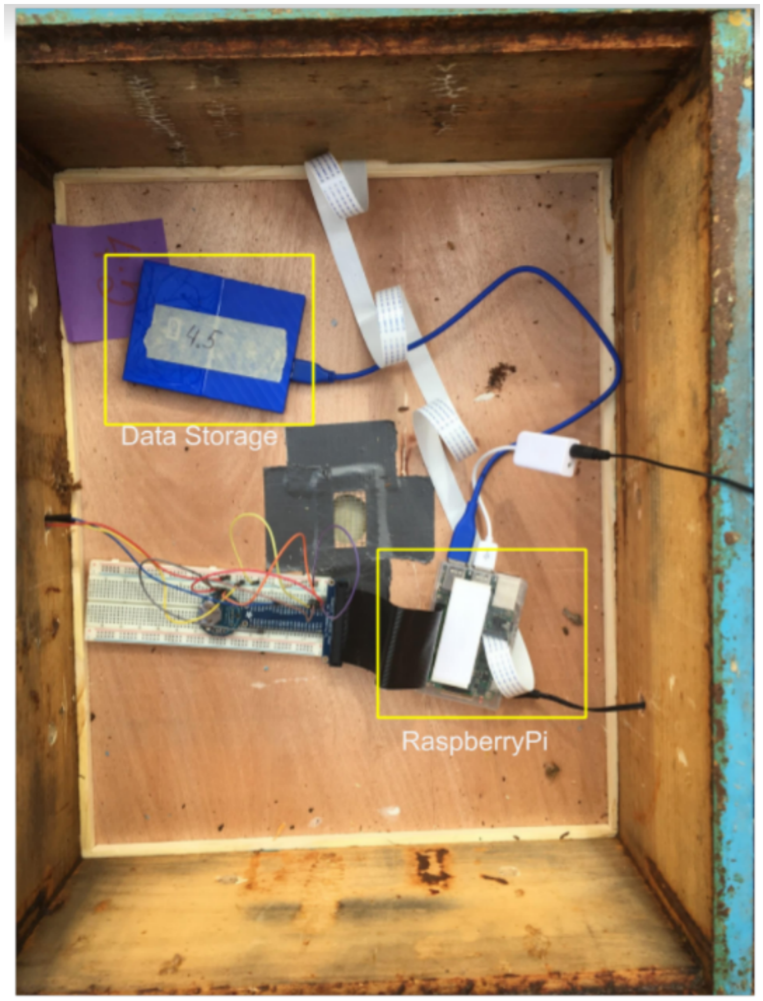

Fig. 2.1: Hardware Components of BeePi

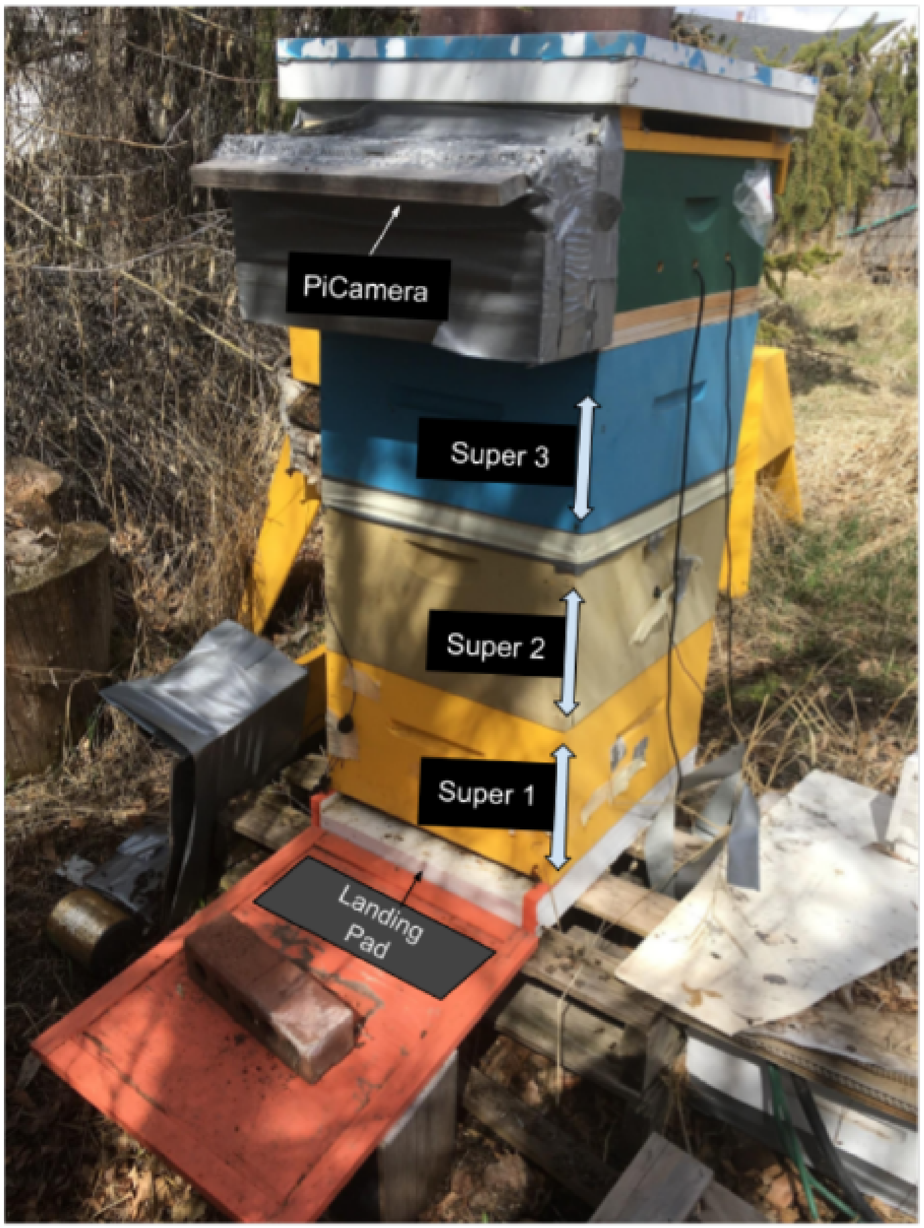

Fig. 2.2: Overall set up of BeePi

The whole setup consists of boxes, we refer to as “supers”. Each box can contain up to five frames. A frame is a rectangular mesh on which the bees build their hives, reproduce and store honey. We need to add more boxes in the future as the population of bees increases.

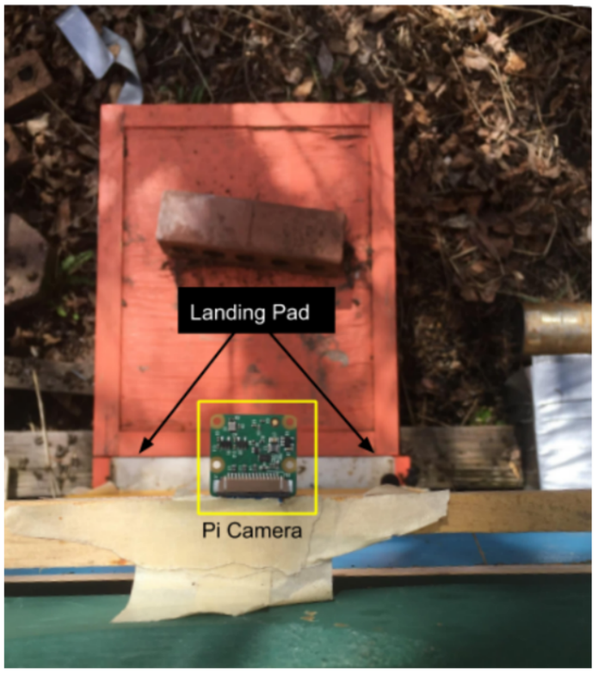

Fig. 2.3: Pi camera mounted on top of hives

As seen in 2.3, Raspberry Pi camera is mounted on the top-most super and facing the landing pad. The distance of the pi camera increases from the landing pad as the number of supers increases. The bees appear smaller when we have two supers compared to bees in one super. All the bee movements over the landing pad are captured by the camera.

## 2.3 Datasets

We have used three different datasets in our thesis. Each dataset is clearly explained in subsequent sections.

### 2.3.1 BEE1

BEE1 dataset consists of 54,391 images of size 32X32 for bees and no-bees. Each video is composed of 749 frames with a resolution of 360x240. 40 videos were randomly selected to generate this dataset. From each video, 400 frames are randomly sampled and were again subjected to random sampling to pick the start and end coordinates of each region. This results in 3980 32x32 images from each video. All these images are pooled to form this dataset. The images were manually labelled into two categories: BEE and NO-BEE. Table 2.1 illustrates the distribution of data for train,test and validation.

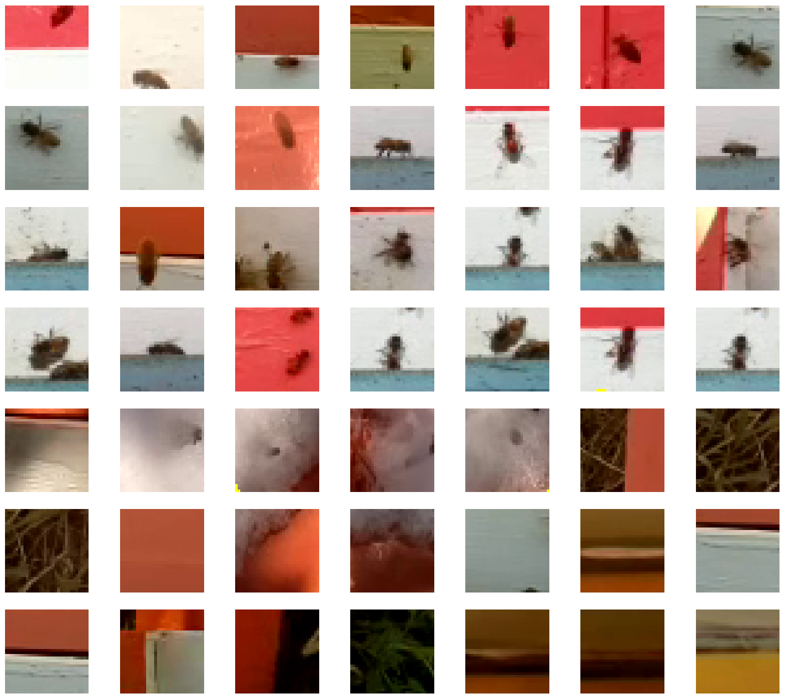

Fig. 2.4: Sample images from BEE1;the first four rows consist images classified as BEE; last three rows of images are classified as NO-BEE

Table 2.1: Number of Train, Test and Validation samples on BEE1 dataset

| BEE1 | Train | Test | Validation | Total |
|---|---|---|---|---|
| Bee | 19,082 | 6362 | 1810 | 27,254 |
| No Bee | 19,057 | 6362 | 1718 | 27,137 |
| Total | 38,139 | 12,724 | 3528 | 54,391 |

### 2.3.2 BEE2

This dataset has 112,879 labelled images. One super(BEE2_1S) dataset have videos that were collected between 05 May and 06 September 2018 and Two Super(BEE2_2S) have videos from 06 September to 07 October 2018. The videos were captured at a resolution of 1920x1080. Each 1-super and 2-super has 50 randomly selected videos. MOG algorithm is used to extract 54,201 150x150 images from 1-super videos and 54,678 90x90 images from 2-super videos. The images were manually labelled into two categories: BEE and NO-BEE to obtain the ground truth classification. Videos from hives 4.5 and 4.7 were used for train and test and 4.8 and 4.10 were used for validation.

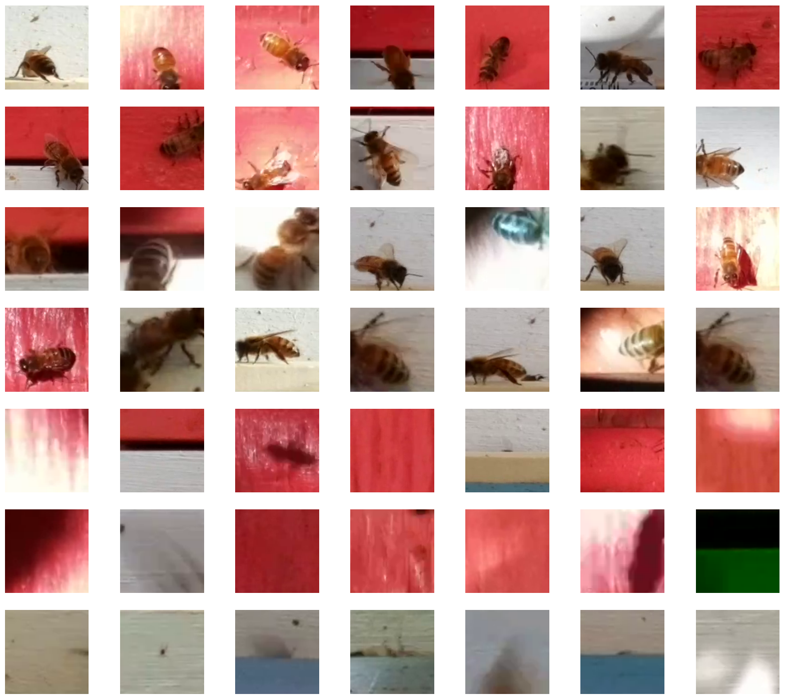

Fig. 2.5: Sample images from BEE2_1S;the first four rows consist images classified as BEE; last three rows of images are classified as NO-BEE

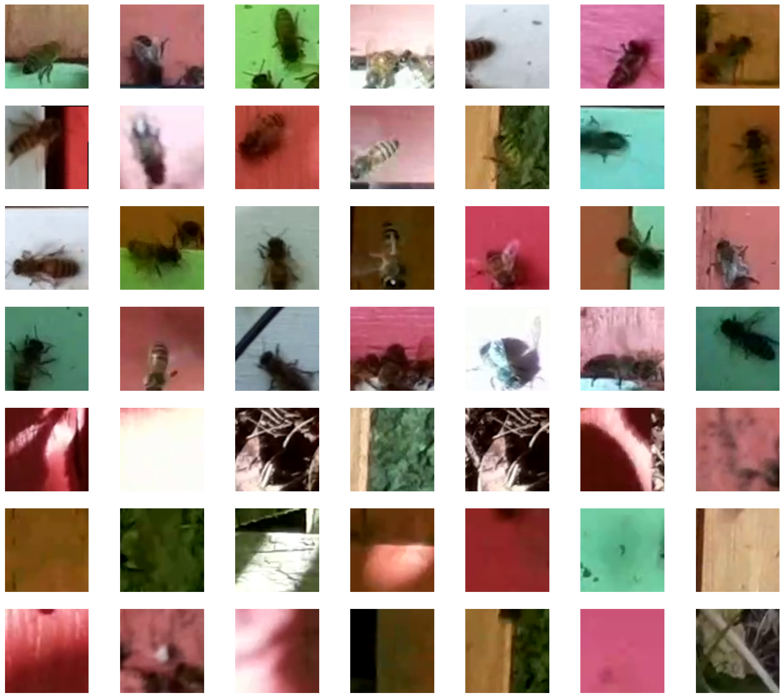

Fig. 2.6: Sample images from BEE2_2S;the first four rows consist images classified as BEE; last three rows of images are classified as NO-BEE

Table 2.2: Number of Train, Test and Validation samples on BEE2 dataset

| | Train | | Test | | Validation | | Total |
|---|---|---|---|---|---|---|---|
| | Bee | No Bee | Bee | No Bee | Bee | No Bee | |
| BEE2_1S | 8266 | 27,108 | 2828 | 9035 | 8298 | 2666 | 58,201 |
| BEE2_2S | 12,982 | 15983 | 4194 | 5327 | 6823 | 9369 | 54,678 |
| Total | 21,248 | 43,091 | 7022 | 14,362 | 15,121 | 12,035 | 112,879 |

### 2.3.3 BEE3

BEE3 dataset has 64,998 64x64 images, obtained from the videos collected between 05 May and 06 June 2018. On an average, each video has 745 frames at a resolution of 1080x1920. We used motion detection algorithms: KNN [22],MOG [23],MOG2 [24] to extract images from the videos. This algorithm detects motion in the current frame by taking previous frame as reference and if it detects a motion, regions of size 64x64 are cropped and saved. This dataset is mostly focused on shadows of the bees. The data is very limited to collect images for shadow bees. So, we took the videos which were captured in the afternoon and manually cropped the shadows of bees from the frames. Shadow bee data is augmented to increase the data for train and test and were resized to a size of 64x64 while training, testing and validating the classifiers. The ground truth classification is obtained by manually labelling the images into three categories: BEE, NO-BEE and SHADOW-BEE. We used videos from hives 4.5 and 4.8 for train and test and videos from 4.7 and 4.10 are used for validation.

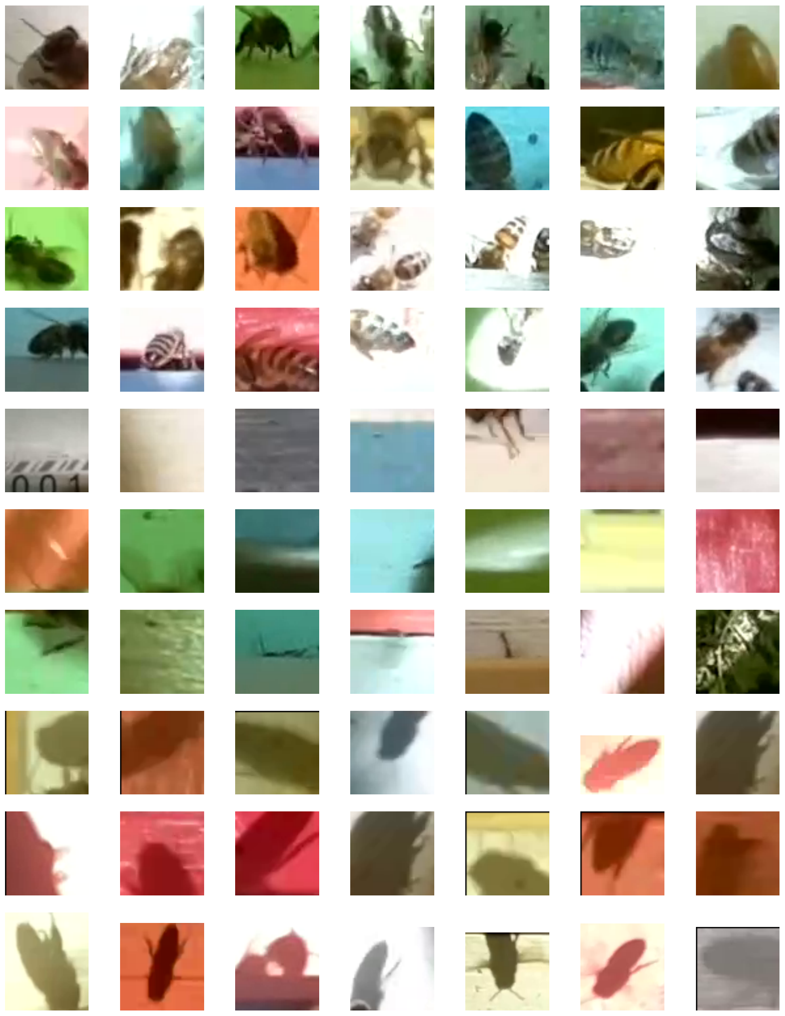

Fig. 2.7: Sample images from BEE3;the first four rows consist images classified as BEE; the next three rows of images are classified as NO-BEE; last three rows of images are categorised as SHADOW-BEE

Table 2.3: Number of Train, Test and Validation samples on BEE3 dataset

| BEE3 | Train | Test | Validation | Total |
|---|---|---|---|---|
| Bee | 12,948 | 4236 | 5187 | 22,371 |
| No Bee | 12,857 | 4242 | 5179 | 22,278 |
| Shadow Bee | 12,006 | 3993 | 4350 | 16,749 |
| Total | 37,811 | 12,471 | 14,716 | 64,998 |

## 2.4 MANOVA

Analysis Of Variance(ANOVA) [25] is a collection of statistical models to determine the significant difference among groups. 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. It is useful for comparing three or more group means for statistical significance. If the observed statistic in the sampling distribution is found to be more than critical value i.e, 0.05, then the null hypothesis is rejected otherwise, it is retained. The PR(¿F) value is responsible to determine significant difference between the groups. Specifically, it tests the null hypothesis:

$$H_0 : \mu_1 = \mu_2 = \mu_3 = ... = \mu_k$$

where $\mu$ is group mean and k is the number of groups.

Multivariate Analysis Of Variance(MANOVA) [26] is merely an ANOVA that has been extended to apply for two or more dependent variables. Image cannot be represented in one variable. So we have chosen MANOVA since it can test the difference in two or more means.

### 2.4.1 MANOVA on Datasets

We extracted three image features - Contrast, Energy and Homogeneity from the GLCM using FEATURE from SKIMAGE library [27]. We labelled the data set (train data as 0, test data as 1 and validation data as 2) and performed MANOVA analysis on each data set. By observing the table 2.4.1, we can say that all the data sets are significantly different from training, testing and validation data sets as their Pr(>F) value is less than 0.05.

Table 2.4: MANOVA analysis for three datasets

| Data set | Df | Pillai | approx F | Pr(>F) |
|---|---|---|---|---|
| BEE1 | 1 | 0.00088021 | 15.971 | 2.244e-10 |
| BEE2_1S | 1 | 0.14443 | 3274.7 | 2.2e-16 |
| BEE2_2S | 1 | 0.021043 | 391.75 | 2.2e-16 |
| BEE3 | 1 | 0.0042428 | 92.309 | 2.2e-16 |

# CHAPTER 3
# SHADOW RECOGNITION TECHNIQUES

## 3.1 OTSU Segmentation

First, we tried image segmentation to detect the shadow pixels of images. Thresholding is a very effective method to separate objects from the background [15]. The proposed algorithm can be seen in 3.1. The input RGB image is converted to HSV color space. Hue(H) refers to the dominant color witnessed by observer. Saturation(S) represents the purity or the amount of light combined with Hue. The areas covered by shadows have maximum value of saturation and minimum value component in this color space. The NDI image is extracted using the S and V (brightness) components. OTSU method of thresholding is used to identify shadow and non-shadow pixels. The obtained image is a binary image. All the shadow and non-shadow pixels are set to 1 and 0 respectively.

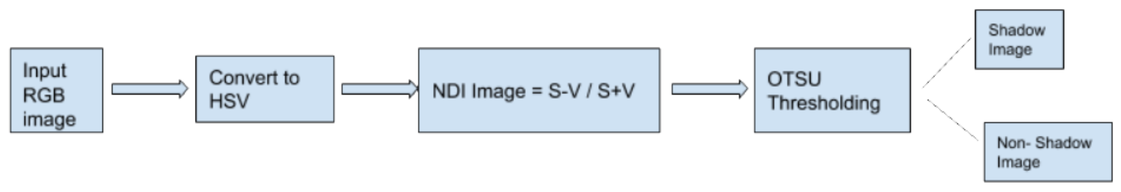

Fig. 3.1: Algorithm for Shadow Segmentation

Now, we tested OTSU thresholding [28] on V (brightness) channel of those color spaces and also Gaussian blurring [29] was added to see whether blurring the RGB images would help to reduce noise.

## 3.2 LAB Color Space

The RGB image is converted to LAB color space. L channel gives the illumination information so it is easy to locate the shadow regions as they are darker than the surroundings. The A channel represents red/green values and B channel represents the blue/yellow values. In most of the outdoor images, the shadow area values for B channel are lesser. As proposed in the paper [14], we computed the mean values of pixels in L,A and B channels of image separately. Then if the mean of A and B channels is greater than $T_1$, all the pixels with a value in L$\leq$ mean(L)-standard deviation (L/3) are classified as shadow pixels or else pixels with value in L+B $\leq T_2$ are also classified as shadow pixels.

## 3.3 Canny Edge Detection

Detecting the shadows based on edges in an image looked convincing. So, we investigated further to check whether the shadow edges and bee edges showed up any difference in their properties.

A OpenCV function Canny [30] was used to get a list of coordinates of the edge pixels. The output of this would be a binary with white pixels along the edge. Those boundary pixel values of the original image are taken and mean is computed. The obtained mean is then thresholded empirically.

# CHAPTER 4
# FEATURE EXTRACTION AND SELECTION

## 4.1 Feature Extraction

Feature extraction simplifies the large set of data to the minimal amount of resources required to perform analysis accurately. Generally, analysing images requires a large amount of memory and computation power. Training the algorithms with such enormous amount of data also over fits the training samples and gives poor performance results. Textural features represents the characteristics of image. Features are computed from the statistical distribution of intensity points(pixels) combinations which are relative to each other at specified positions in the image. The pixels in each combination fall into first-order, second-order and higher-order statistics.

We are using a popular statistical method of extracting features from image which is Gray Level Co-ocurrence Matrix (GLCM) [16]. GLCM tabulates the information about the positions of pixel having similar gray level values. It represents the relationship between two pixels which are known as reference and neighbor pixel. This is as square matrix with $N_g$ dimension, where $N_g$ is the number of gray levels in the image and both rows and columns representing a set of image values. Each element [i,j] of the matrix is obtained by the number of occurrences of the pixel with value i appeared with a pixel with value j. The matrix element P(i,j$\|, d$,$\theta$) has the probability values for changes between i and j at a displacement distance d at an angle $\theta$.

To compute GLCM, we used GREYCOMATRIX implemented from SKIMAGE library. The feature values for the given GLCM are extracted by using the following formulas [31]:

1. Angular Second Moment(ASM)

$$\sum_i \sum_j P(i,j)^2$$

2. Contrast

$$\sum_i \sum_j (i-j)^2 P(i,j)$$

3. Correlation

$$\sum_i \sum_j \frac{(ij)P(i,j) - \mu_x \mu_y}{\sigma_x \sigma_y}$$

where $\mu_x$,$\mu_y$,$\sigma_x$ and $\sigma_y$ are the means and std.deviations of $P_x$ and $P_y$, the partial probability density functions.

4. Sum of Squares: Variance

$$\sum_i \sum_j (i-\mu)^2 P(i,j)$$

where $\mu$ is the mean of GLCM matrix.

5. Inverse Difference Moment(IDM)

$$\sum_i \sum_j \frac{P(i,j)}{1+(i-j)^2}$$

6. Sum Average(SA)

$$\sum_{i=2}^{2N_g} i P_{x+y}(i)$$

7. Sum Variance

$$\sum_{i=2}^{2N_g} (i-SA)^2 P_{x+y}(i)$$

8. Sum Entropy

$$-\sum_{i=2}^{2N_g} i\ log\{P_{x+y}(i)\}$$

9. Entropy

$$-\sum_i \sum_j P(i,j)\ log\{P(i,j)\}$$

10. Difference Variance

$$\sum_i i^2 \; P_{x-y}(i)$$

11. Difference Entropy

$$-\sum_i P_{x-y}(i) \; log\{P_{x-y}(i)\}$$

12. Info. Measure of Correlation 1

$$\frac{HXY - HXY1}{max\{HX, HY\}}$$

13. Info. Measure of Correlation 2

$$\Big(1 - exp[-2(HXY2 - HXY)]\Big)^{1/2}$$

where HXY is entropy, HX, HY are the entropy's of $P_x$ and $P_y$ and
$HXY1 = -\sum_i \sum_j P(i,j) \; log\{P_x(i)P_y(j)\}$ $\;HXY2 = -\sum_i \sum_j P_x(i)P_y(j) \; log\{P_x(i)P_y(j)\}$.

14. Auto Correlation

$$\sum_i \sum_j (ij) \; P(i,j)$$

15. Dissimilarity

$$\sum_i \sum_j abs(i-j) \; P(i,j)$$

16. Homogienity

$$\sum_i \sum_j \frac{P(i,j)}{(1+(i-j)^2)^2}$$

17. Inverse Difference Normalized(IDN)

$$\sum_i \sum_j \frac{P(i,j)}{1 + abs(i-j)/N_g}$$

18. Maximum Probability is the maximum value of GLCM matrix.

19. Cluster Shade

$$\sum_i \sum_j (i + j - \mu_x - \mu_y)^3 \, P(i,j)$$

20. Cluster Prominence

$$\sum_i \sum_j (i + j - \mu_x - \mu_y)^4 \, P(i,j)$$

21. Energy

$$\sqrt{ASM}$$

### 4.1.1 Principal Component Analysis

Principal Component Analysis(PCA) is a dimensionality reduction technique that transforms a number of correlated variables into a number of linearly uncorrelated variables called principal components. It involves substantial number of correlated variables for multivariate analysis. Principal components aims at maintaining most of the information from original data. The first principal component possess the utmost variance of the dataset and each of the following component accounts for the remaining variance. The resulting components are an uncorrelated orthogonal basis set i.e. they are statistically independent to each other [32]. PCA is performed on a square symmetric matrix that explains how our variables relate to one another. The matrix is broken down into two components: direction and magnitude. Direction is the combination of data and magnitude indicates the importance of each direction.

## 4.2 Feature Selection

After extracting features from the images, it is important to analyze the significant features which helps the model for better classification rather than creating noise. The process of selecting subset of relevant features is called as feature selection. This reduces overfitting, improves the accuracy by mostly eliminating misleading data and also helps to reduce the complexity of the model to train faster. In order to get optimal subset of features, most of the feature selection techniques scans the entire feature set and ranks all the features based on evaluation metrics. This section describes some of the feature selection techniques we used.

### 4.2.1 Sequential Feature Selection

Sequential Feature Selection belongs to wrapper methods. These methods use an extensive greedy search algorithm to find the best subset of features which produces best result for a specific model. SFAs adds or removes one feature at a time based on the performance of the model to reduce the dimensions in the feature space. The number of features required must be chosen empirically. The algorithm stops when the given number is reached. SFAs are categorised into two groups:

- Sequential Forward Selection (SFS): Initially, they train and evaluate $n$ classifiers (n is the dimension of feature set) with respect to each feature. The feature that gave the best performance towards classification is selected. Again, n-1 classifiers are trained and evaluated with the first feature. The combination of two features that yield best performance is selected. The process stops when the specified number of features is selected.

- Sequential Backward Selection: In this process, one feature is removed at a time from the feature set and performance is evaluated by the classifier. The feature set that yield best results towards classification is retained. The process stops when the specified number of features is selected.

### 4.2.2 Univariate Feature Selection

Univariate Feature Selection (UFS) comes under Filter methods. This technique analyzes each feature to determine the strength of the relationship of the feature with target variable. The rank of the features are determined by the strength of their relationship with the outcome based on some statistical tests such as F-test, Mutual Information, chi-square, etc. Except a preset number of highest scorers, all other features are removed from the feature space. The rest of the features are used to train and evaluate the classifiers. In our work, we used F-test and Mutual Information univariate statistical tests.

- F-test: It is a statistical test used to compare between the feature and target variable to check if the difference is significant. F-test captures the relationships between the features and target variables. Correlation scores(F-score) are computed for the features. The features with higher F-scores are selected for modelling.

- Mutual Information: This measures the dependency between target variable and the feature variable. The mutual information value is 0, when the target is independent of the feature. It is 1, when the target is fully dependent on the feature. In other cases, the values ranges between 0 and 1. The features with higher mutual information values are selected for modelling.

# CHAPTER 5
# CLASSIFICATION MODELS

Over last few years, Deep Learning (DL) has given rise to a massive collection of concepts that were previously fragmented and disparate. DL methods have appeared to outperform previous state of the art Machine Learning(ML) methods in several fields, noticeably standing out in the field of computer vision. Conventional ML techniques are restricted to process raw data. For a long time, ML techniques needed lot of domain expertise, human intervention and careful feature engineering transformed raw data into a feature vector from which a learning system such as classifier could detect or classify patterns in the input. However, ML is highly susceptible to errors when the data is covered by irrelevant features. The goal behind DL is to automate the process of learning features from data; here the algorithms model high level abstractions i.e. they try to learn high-level features from data in an incremental manner using architectures to produce the output accurately.

The next sections provide detailed explanations of DL models such as Convolution Neural Networks and State of the art models and Machine Learning models such as Random Forest and SVMs.

## 5.1 Convolution Neural Networks

ConvNets are a derivative of standard MLP neural networks optimised for two-dimensional pattern recognition problems. ConvNets process the data that come in the form of multiple arrays. It is a good answer for multiple trainable stages stacked top on each other. Feature maps which are sets of arrays, generated at each stage. Particular features are extracted at all locations on the input which represents each feature map. ConvNets have neurons arranged in 3 dimensions (width,height,depth). Neurons in feature maps receives input from a receptive field which is a restricted subarea of previous layer. Through set of weights called filter bank, neurons are connected to local patches in feature maps of previous layers.

In a feature map, all the neurons share the same filter bank. Different filter banks are used by different feature maps. The result of this sum is passed through a non-linearity such as ReLU. This kind of architecture for images forms distinctive local motifs, high correlation and local statistics are invariant to location. They include pooling for data reduction called as sub-sampling operations. This operation is applied to the output of previous layer. Pooling computes maximum value of a predefined window in a feature map by eliminating the non-maximal values. The output layer classifies the input image, every neuron in this layer is fully connected to the neurons in previous layer. We stack different types layers to form a full ConvNet architecture [33, 34]. Each layer type is described below sections.

### Convolution Layer

In Convolution Layers, a convolution operation is applied to the input from previous layers and the filters. It produces $n$ feature maps, where $n$ is the number of filters and pass on to the next layer. The main function of convolution layer is to detect the local conjunctions of features from the previous layer and mapping their appearance to a feature map. The ConvNet learns the filters which get activated when a specific feature is detected. Most of the neurons share same vector of weights and bias. The number of filters used to convolve the input results in the same amount of feature maps which can be seen in figure 5.1.

### Activation Function

Activation function is a key component for a neural network. These non-linear activation functions are transfer functions that convert the linear inputs to non-linear outputs. The function of the activation function depends on the position it is placed. If it is placed after the hidden layers, then it transforms the linear mappings to non-linear for propagation. If placed in the output layer, it performs predictions or classifications [35].

1. **Softmax**: We have used Softmax as the activation function in the output layer for all the architectures. It takes a vector of real numbers and calculates the probability

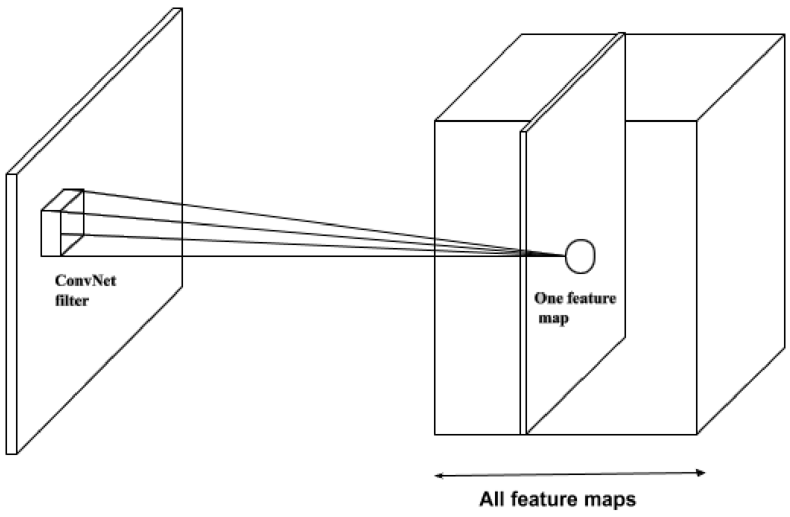

Fig. 5.1: Feature map with one filter

distribution. The output of this activation function varies between 0 and 1. Softmax activation function is used for multivariate classification models and the sum of probabilities of each class equals to 1.

$$f(x) = \left( \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}} \right) \tag{5.1}$$

where $i$ = 1,2,...,J.

2. **ReLU**: The role of Rectified Linear Unit(ReLU) layer is to saturate or limit the generated output. ReLU favours for training because it creates a sparser representation, which means zero in the gradient leads to a complete zero unlike sigmoid or tanh and also for positive input, it has constant gradient [35, 36]. The linear function and gradient are defined as:

$$ReLU(x) = max(0, x)$$

$$\frac{d}{dx}(x) = \{1 \text{ if } x > 0;\ 0 \text{ otherwise}\} \tag{5.2}$$

3. **Tanh**: Hyperbolic Tangent Function(Tanh) helps the back-propagation process to produce zero centered output. It is a smoother function whose values varies between

-1 to 1 [35]. The output of this function is as follows:

$$f(x) = \left(\frac{e^x - e^{-x}}{e^x + e^{-x}}\right) \tag{5.3}$$

### Pooling Layer

Pooling layer task is to combine features which are semantically similar to one. It is down-sampling the dimensions of data. Pooling function replaces the output of previous layer at specific location with one value by computing statistics on nearby outputs. Pooling does not affect the number of filters and can be used with non-equal filters. If we pool the input by a small amount, most of the pooled outputs do not change. Convolution or non-linearity layers are usually succeeded by pooling layers. Max-pooling is one of the pooling methods. It returns the maximum value from the sub-regions of an image. Pooling operation does not preserve the information of a position [36].

### Dropout

This drops out units at each training step in a neural network. It temporarily removes the units from all incoming and outgoing connections. Either the probability is kept 1-p or p. This prevents units from co-adapting too much. Applying dropout results to sampling a "thinned" network. The thinned network consists of all units that survived the dropout. A neural network with $n$, can be seen as a collection of $2^n$ possible thinned neural networks [37].

### Batch Normalization

Batch Normalization reduces the amount of internal covariance shift. It allows each layer in the network to to learn a bit more independently by itself. High learning rates can be used without any risk of divergence. It also regularizes the model and reduces the use of dropout. We additionally add this normalization for convolution layers, so that it obeys the convolution properties and different elements of feature map, at different locations are normalized in the same manner [38]. We can apply Batch Normalization to any set of activations. First, affine transformation is performed and then element-wise nonlinearity:

$$z = g(Wu + b)$$

where $W$ and $b$ are learned parameters of the model and *g(.)* is nonlinearity such like sigmoid, tanh or ReLU. Batch Normalization transform is applied before the nonlinearity, by normalizing *(Wu+b)*. The Batch Normalization formula used is:

$$y^{(k)} = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)}$$

where $x^{(k)}$ is for each activation and $\gamma^{(k)}$ ,$\beta^{(k)}$ are a pair of parameters, by setting $\gamma^{(k)} = \sqrt{Var}[\mathrm{x}^{(k)}] and \gamma^{(k)} = \mathrm{E}[x^{(k)}]$.

**Fully Connected Layer**

Each neuron in a fully-connected layer is connected to every neuron in both the preceded and succeeded layers. It is better to eliminate the number of neurons and connections in this layer because the computation complexity increases as per the number of parameters. The weight parameters are tuned based on the output activation maps, obtained by concatenating the convolution, non-linearity and pooling layers to create stochastic likelihood of each class.

The internal architecture of a Convolution Neural Network is shown in figure 5.2. It has convolution layer which results in a feature map, the obtained feature maps are downsampled in pooling layer and again a convolution layer resulting in feature maps. Next, is the fully connected layer which classifies the input.

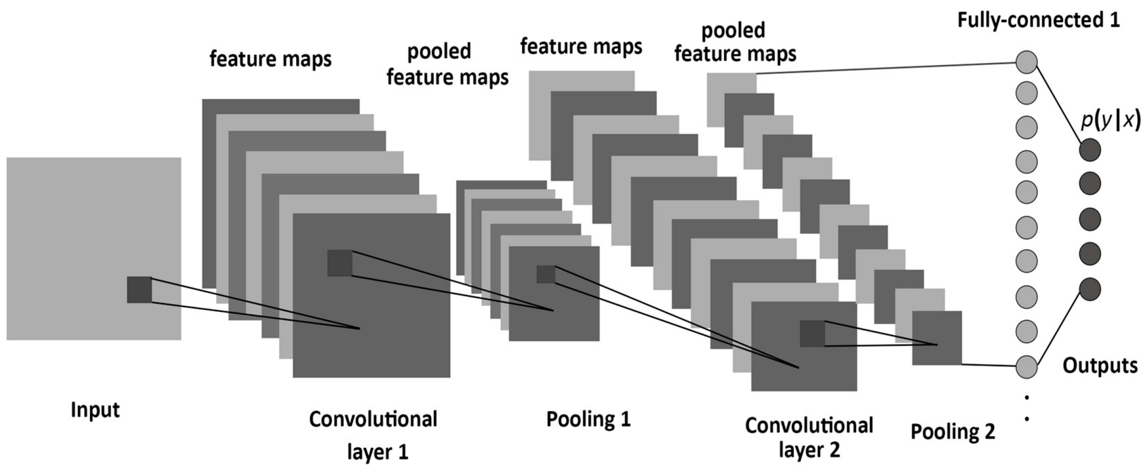

Fig. 5.2: CNN Architecture [1]

## 5.2 State of the art models

### VGGNet16

The detailed description and control flow of this model is shown in 5.3 This network is comprised of convolutional layers with 3x3 receptive field filters and stride fixed to 1 and max-pooling layers with kernel size 2 and stride 2. Followed by 3 FC layers, the first two of them have 4096 units each and the third is the softmax layer which performs classification with the desired number of output classes.ReLu is the activation in all the layers. In our experiments, we have used rmsprop as optimizer [39].

### VGGNet19

This model is very much similar to the previous model. Three more convolutional layers and a flattened layer are added to the architecture. We have used adam optimizer. The layer by layer control flow is shown in Fig.5.4

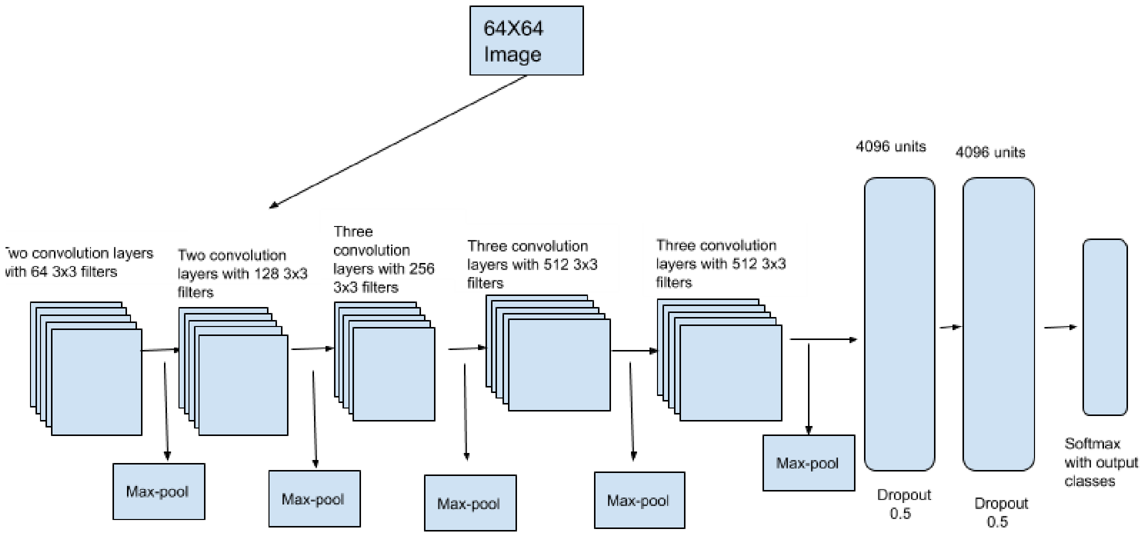
64X64 Image
Two convolution layers with 64 3x3 filters
Two convolution layers with 128 3x3 filters
Three convolution layers with 256 3x3 filters
Three convolution layers with 512 3x3 filters
Three convolution layers with 512 3x3 filters
Max-pool
Max-pool
Max-pool
Max-pool
Max-pool
4096 units
4096 units
Dropout 0.5
Dropout 0.5
Softmax with output classes

Fig. 5.3: VGGNet Architecture with 16 weighted layers

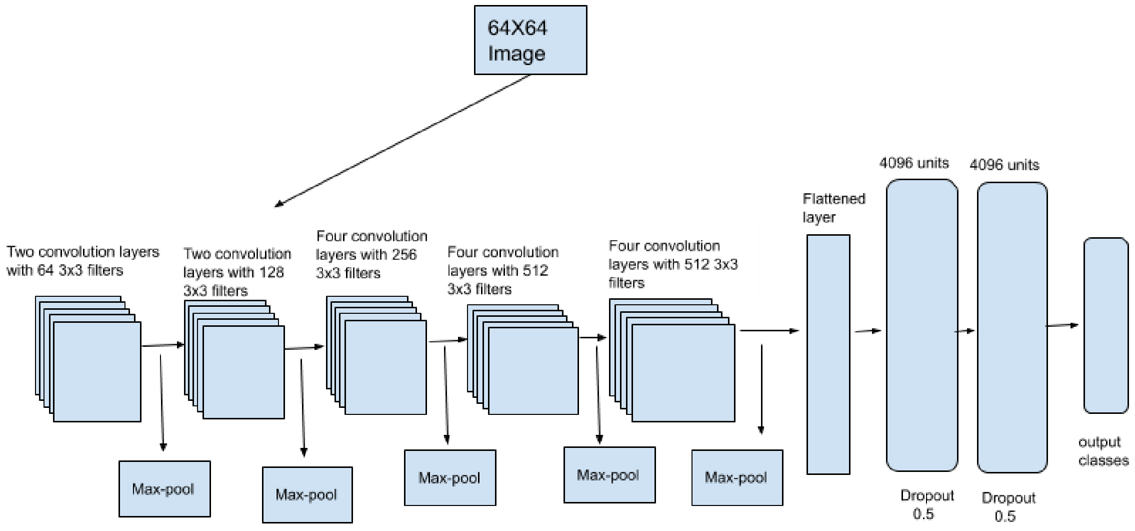
64X64 Image
Two convolution layers with 64 3x3 filters
Two convolution layers with 128 3x3 filters
Four convolution layers with 256 3x3 filters
Four convolution layers with 512 3x3 filters
Four convolution layers with 512 3x3 filters
Max-pool
Max-pool
Max-pool
Max-pool
Max-pool
Flattened layer
4096 units
4096 units
Dropout 0.5
Dropout 0.5
output classes

Fig. 5.4: VGGNet Architecture with 19 weighted layers

### ResNet

ResNet [19] architecture has 32 layers along with convolutional layers, max-pooling layers, Residual blocks and Normalization layers. Residual blocks as shown in Fig.5.5 are used to address the degradation problem. Each convolutional layer is followed by ReLU. The optimizer used is Momemtum. The architecture of this model appears as Fig.5.6

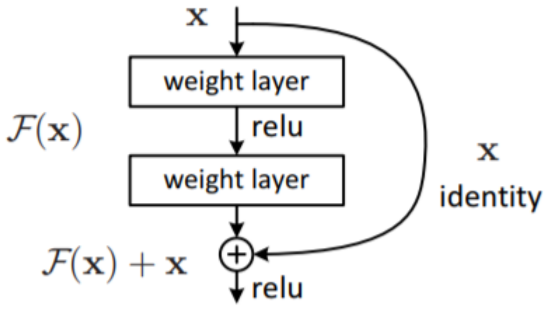
x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Fig. 5.5: Residual Learning building block

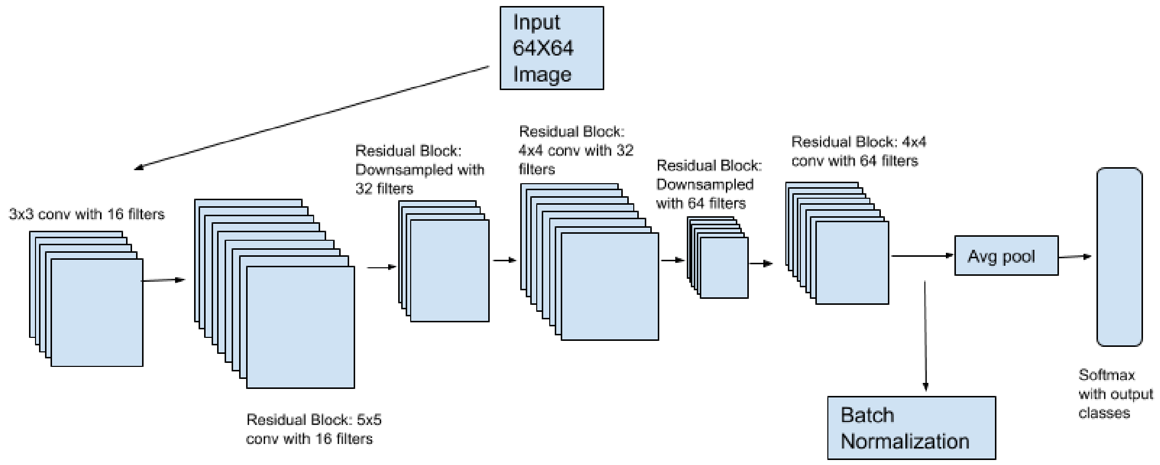
Input 64X64 Image
3x3 conv with 16 filters
Residual Block: 5x5 conv with 16 filters
Residual Block: Downsampled with 32 filters
Residual Block: 4x4 conv with 32 filters
Residual Block: Downsampled with 64 filters
Residual Block: 4x4 conv with 64 filters
Avg pool
Batch Normalization
Softmax with output classes

Fig. 5.6: ResNet Architecture

## AlexNet

AlexNet [18] is comprised of convolutional with different filters and filter sizes, max-pool layers with kernel size 3 and stride 2, local response normalization and FC layers. The ReLU is used as the activation for all convolution layers, tanh for FC layers and softmax for the output layer and momentum is the optimizer. The architecture of this model is clearly shown in Fig.5.7.

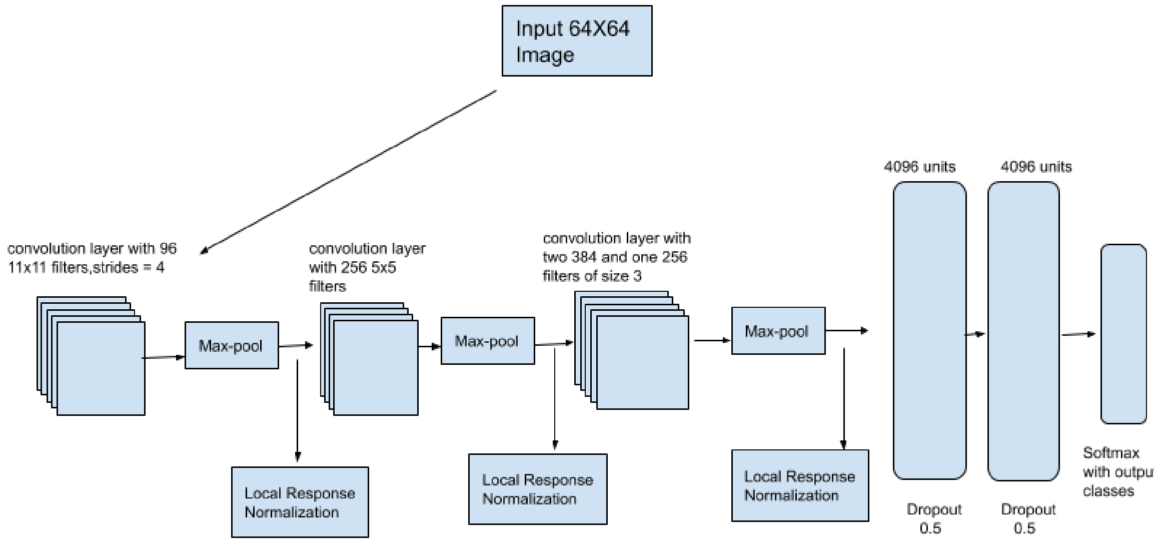
Input 64X64 Image
convolution layer with 96 11x11 filters,strides = 4
Max-pool
convolution layer with 256 5x5 filters
Local Response Normalization
Max-pool
convolution layer with two 384 and one 256 filters of size 3
Local Response Normalization
Max-pool
Local Response Normalization
4096 units
4096 units
Dropout 0.5
Dropout 0.5
Softmax with outpu classes

Fig. 5.7: AlexNet Architecture

### ZFNet

ZFNet architecture is close to AlexNet with some minor differences. The first convolution layer have 96 7x7 filters with strides as 2. The second convolution layer is modified with the number of strides and rest of the convolution layers have different filters. Detailed structure of architecture can be seen in Fig.5.8

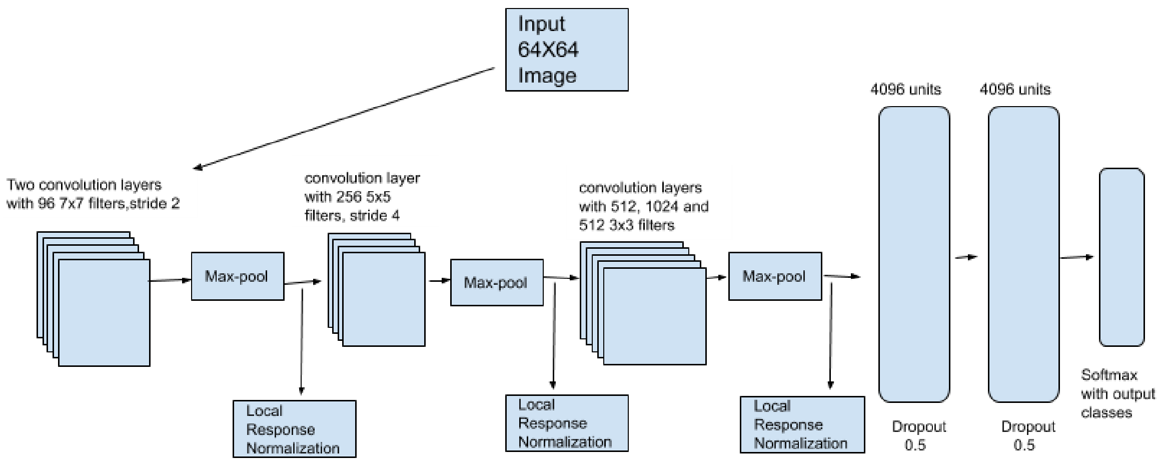

Fig. 5.8: ZFNet Architecture

**GoogleLeNet**

GoogleLeNet [2] incarnates the architecture of Inception. Fig.5.9 shows the inception module with dimensionality reduction. This architecture has 9 inception modules, containing 22 layers along with four max-pooling layers and one average pooling layers. Including the inception modules, ReLU is used in all the convolutional layers. A 0.4 dropout ratio is applied to the softmax layer and Momentum is the optimizer. We can see the control flow of this architecture in Fig.5.10.

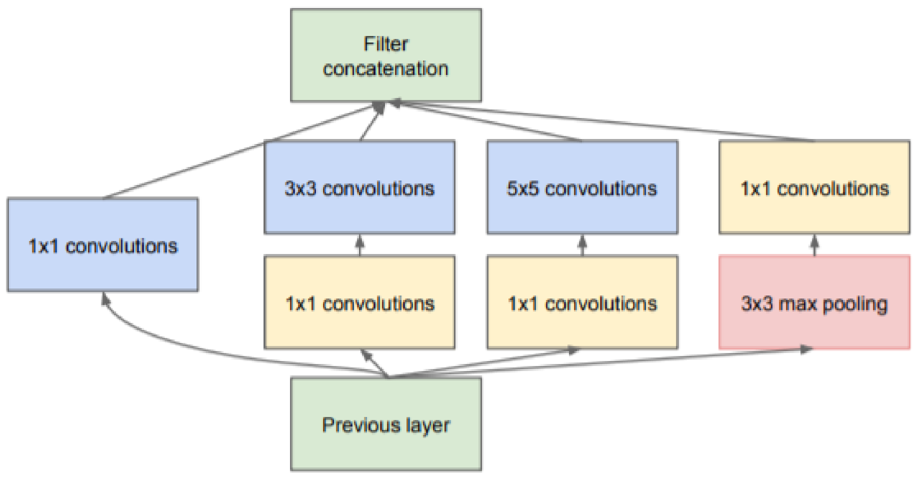

Fig. 5.9: Inception module [2]

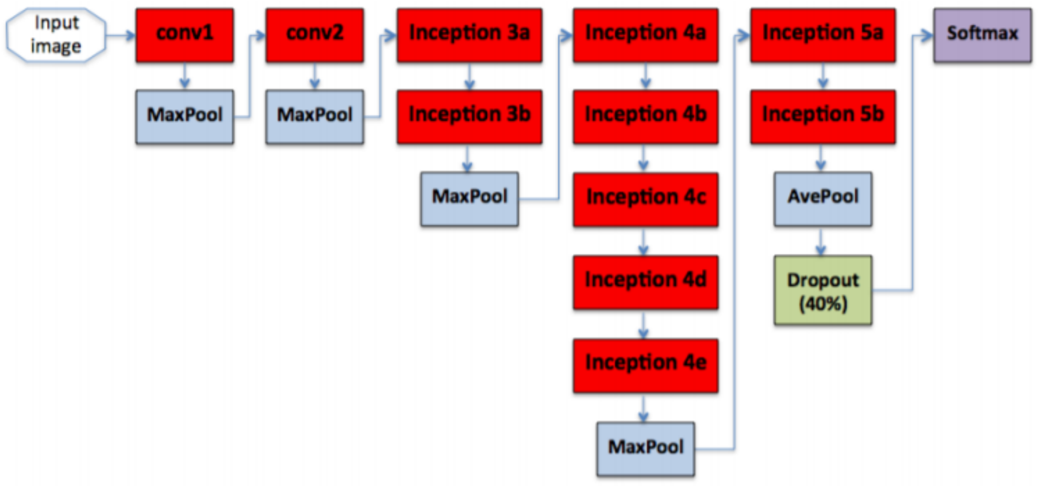

Fig. 5.10: GoogleLenet Architecture [3]

## 5.3 Machine Learning

Machine learning (ML) is based on algorithms that have the capability to learn from data without being explicitly programmed. The learning process depends on the patterns and inference of the input. Supervised Learning is a Machine Learning task of learning a function that maps an input to an output. The training data consist of a set of training examples and each example is a pair of input object(typically a vector) and a desired output value. The learning algorithm predicts the training examples iteratively and gets corrected by the training examples. The learning stops once the algorithm achieves an acceptable level of performance. The function inferred from training examples can be used later for predicting the output value for new examples. Supervised Learning is grouped into Regression and Classification. Regression may predict a continuous quantity or size and can be evaluated using root mean squared error, whereas Classification may predict a discrete class label and can be evaluated using accuracy.

We have used Support Vector Machine (SVM) and Random Forest (RF) ML techniques in our experiments as they were well-used models for image classification.

### 5.3.1 Random Forest

Random Forest is an ensemble classification method used for classification and regression. The basic building block of Random Forest is a Decision Tree. The“forest” it builds, is a collection of Decision Trees. Using “bagging” method each tree considers a subset of random features sampled independently from the training data. Instances which appear at least once in the sample are in-bag instances, and the other instances are out-of-bag instances. It make predictions by averaging over the predictions made by several decision trees. Each tree in the forest votes for the class label of an unlabelled instance. Then by majority voting, the class label with most votes classifies the instance. It merges the predictions to increase overall accuracy of the result.

Random Forest error rates depends on Correlation and Strength. The error rate decreases statistically by maximizing the strength of trees and minimizing the correlation between trees.

**Variable Importance**

RF searches for the best feature among a random subset of features while splitting the tree. The sum of importance of all variables is equal to 1. The importance of each variable in RF can be measured in two ways.

1. **Accuracy-based Importance**: The out-of-bag sample of each tree is used to calculate the importance of variables. Initially, the prediction accuracy is measured and then the values of variables are shuffled randomly as the shuffled variables has no predictive power. The mean decrease in the prediction accuracy on that data gives the variable importance measure.

2. **Gini-based Importance**: Gini impurity is used to decide on which variable to split at each internal node. Each branch is the outcome of this split. A leaf node represents the class label. The sum of gini decreases every time the same variable is chosen to split across the trees for each variable. The average of this sum is computed by dividing with the number of trees. Minimal computation is required for this because the Gini calculations are already performed during training.

### 5.3.2 SVM

SVM is a supervised ML algorithm used for Classification and Regression. SVM maps the input vector to an n-dimensional space to find the hyperplane that classifies the data points distinctly. The data points of two classes can be separated by several number of hyperplanes. But the best hyperplane that has maximum distance between the nearest data points(support vectors) of both the classes is chosen. These support vectors influence the position and orientation of hyperplane. SVM training algorithm builds a model to classify the new examples by assigning data points to one of the classes. In non-Linear classification problems, SVM uses kernel trick to transform the input space to a high dimensional space to separate the classes by a clear gap that is as wide as possible. In such case, SVM uses soft margin which involves slack variables and a penalty parameter C. The dimension of the hyperplane rely on the number of features [40]

SVM One-Vs-Rest (OVR) builds a linear classifier for data points belonging to N different classes. This approach combines n binary classifiers to solve n class classification. It constructs a function that predicts each one of binary classifiers and then outputs the largest(most positive) value [41].

# CHAPTER 6
# EXPERIMENTS AND RESULTS

## 6.1 Overview

To estimate bee traffic, we need to find the best model that would detect the actual "BEE" from other unwanted information such as "NO BEE" and "SHADOW BEE". So, we trained and evaluated several neural networks, SVM and Random Forests. This section presents the performance results on different shadow recognition techniques, Deep Learning models, Machine Learning and State of the art models on BEE1, BEE2 and BEE3 datasets.

## 6.2 Shadow Recognition Techniques

We tested on bee and shadow bee images in BEE3 dataset. Table 6.1 shows the results of different techniques with their respective threshold values (T). The threshold values used are dataset specific. We selected them empirically. We tested OTSU thresholding on V channel and also added Gaussian blurring to the images with a Gaussian kernel of size 7x7 as the width and height of the kernel should be odd. Both these methods have same performance as OTSU segmentation. We can say that, the Canny edge detector method did not work well with T=160 when compared to T=150. We tried the other method with different $T_1$ and $T_2$ values but they did not show good results. In feature extraction method, we extracted the features of images and saved them in csv files. Then we trained a random forest classifier with 50 trees using PCA components. This method gave us better results. We plotted a graph 6.1 on these techniques which gave higher accuracy with a particular threshold value.

Table 6.1: Test and Validation accuracies

| Technique | Test Accuracy | Validation Accuracy |
|---|---|---|
| OTSU Segmentation | 48.52 | 45.61 |
| OTSU on V | 48.52 | 45.61 |
| Gaussian Blur & OTSU | 48.52 | 45.61 |
| Canny(T=150) | 25.93 | 30.13 |
| Canny(T=160) | 18.32 | 23.68 |
| LAB($T_1 = 256, T_2 = 75$) | 49.11 | 36.04 |
| LAB($T_1 = 256, T_2 = 100$) | 47.19 | 28.68 |
| LAB($T_1 = 100, T_2 = 75$) | 49.11 | 36.04 |
| LAB($T_1 = 75, T_2 = 25$) | 35.38 | 32.05 |
| Feature Extraction | 96.17 | 86.63 |

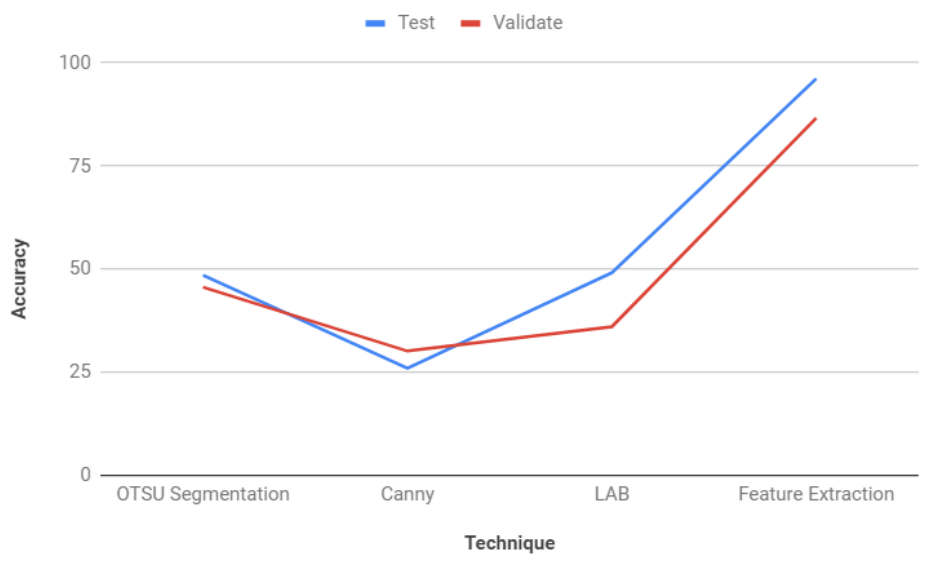

Fig. 6.1: Comparison of all techniques

## 6.3 Feature Extraction

We performed feature extraction technique on all the three datasets. To see the effect of image size, we experimented this technique by doubling the original size of the image each time. By observing the results on BEE1 in Table 6.2, the validation accuracy is increased by 1% which is not significant. On BEE2 OneSuper in Table 6.3, we can see a 2% increase in the validation accuracy when the image is resized to 128x128 pixels using Random Forest classifier. SVMs performed well for this dataset, a significant difference of almost 15% can be seen. On BEE2 TwoSuper in Table 6.4, a 3% increase in validation accuracy is noticed for 128x128 images. Further increasing the size of the image is a bad idea as we can clearly see that the validation accuracy is decreased. Now on BEE3 in Table 6.5, the validation accuracy is increased by 3% for 128x128 images and for this dataset, going beyond 100 trees is decreasing the accuracy of the model.

Table 6.2: Results after Feature Extraction on BEE1 dataset with different sizes

| Model | Test Acc (32x32) | Val Acc (32x32) | Test Acc (64x64) | Val Acc (64x64) |
|---|---|---|---|---|
| RF(50) | 98.16 | 96.45 | 98.33 | 97.39 |
| RF(100) | 98.15 | 96.48 | 98.38 | 97.27 |
| RF(150) | 98.20 | 96.54 | 98.42 | 97.13 |
| RF(200) | 98.25 | 96.57 | 98.43 | 97.22 |
| RF(250) | 98.23 | 96.48 | 98.43 | 97.05 |
| SVM | 81.29 | 71.85 | 79.40 | 70.89 |

Table 6.3: Results after Feature Extraction on BEE2 OneSuper dataset with different sizes

| Model | Test Acc (64x64) | Val Acc (64x64) | Test Acc (128x128) | Val Acc (128x128) | Test Acc (256x256) | Val Acc (256x256) |
|---|---|---|---|---|---|---|
| RF(50) | 90.17 | 59.50 | 90.30 | 61.76 | 90.39 | 61.66 |
| RF(100) | 90.39 | 59.75 | 90.25 | 61.96 | 90.44 | 62.12 |
| RF(150) | 90.39 | 60.27 | 90.32 | 62.25 | 90.51 | 62.24 |
| RF(200) | 90.49 | 60.37 | 90.30 | 62.26 | 90.53 | 62.59 |
| RF(250) | 90.41 | 60.37 | 90.21 | 62.39 | 90.49 | 62.79 |
| RF(300) | 90.44 | 60.30 | 90.29 | 62.28 | 90.45 | 62.65 |
| SVM | 66.24 | 75.44 | 64.34 | 78.80 | 67.88 | 68.84 |

Table 6.4: Results after Feature Extraction on BEE2 TwoSuper dataset with different sizes

| Model | Test Acc (64x64) | Val Acc (64x64) | Test Acc (128x128) | Val Acc (128x128) | Test Acc (256x256) | Val Acc (256x256) |
|---|---|---|---|---|---|---|
| RF(50) | 95.12 | 66.34 | 95.13 | 69.22 | 95.55 | 68.39 |
| RF(100) | 95.18 | 66.56 | 95.21 | 69.54 | 95.58 | 68.88 |
| RF(150) | 95.22 | 66.56 | 95.17 | 69.09 | 95.63 | 69.04 |
| RF(200) | 95.24 | 66.70 | 95.20 | 69.16 | 95.58 | 68.95 |
| RF(250) | 95.30 | 66.69 | 95.24 | 69.33 | 95.63 | 68.90 |
| RF(300) | 95.32 | 66.55 | 95.28 | 69.31 | 95.69 | 69 |
| SVM | 58.90 | 59.93 | 63.62 | 54.77 | 64.08 | 54.74 |

Table 6.5: Results after Feature Extraction on BEE3 dataset with different sizes

| Model | Test Acc (64x64) | Val Acc (64x64) | Test Acc (128x128) | Val Acc (128x128) |
|---|---|---|---|---|
| RF(50) | 85.84 | 71.73 | 86.68 | 73.88 |
| RF(100) | 86.02 | 71.85 | 86.78 | 74.08 |
| RF(150) | 85.95 | 71.83 | 86.76 | 74.02 |
| SVM | 48.71 | 52.0 | 41.71 | 44.20 |

### 6.3.1 Principle Component Analysis

Table 6.6: Results after applying PCA on features on BEE1 dataset with different sizes

| Model | Test Acc (32x32) | Val Acc (32x32) | Test Acc (64x64) | Val Acc (64x64) |
|---|---|---|---|---|
| RF(50) | 98.09 | 96.37 | 98.31 | 96.65 |
| RF(100) | 98.14 | 96.42 | 98.37 | 96.65 |
| SVM | 78.56 | 67.85 | 49.53 | 55.18 |

Table 6.7: Results after applying PCA on features on BEE2 OneSuper dataset with different sizes

| Model | Test Acc (64x64) | Val Acc (64x64) | Test Acc (128x128) | Val Acc (128x128) | Test Acc (256x256) | Val Acc (256x256) |
|---|---|---|---|---|---|---|
| RF(50) | 91.14 | 60.10 | 91.08 | 62.03 | 91.01 | 63.38 |
| RF(100) | 91.15 | 59.53 | 91.29 | 62.65 | 91.13 | 64.32 |
| RF(150) | 91.23 | 59.72 | 91.30 | 62.67 | 91.25 | 64.43 |
| SVM | 72.29 | 35.11 | 46.78 | 68.22 | 39.25 | 66.39 |

Table 6.8: Results after applying PCA on features on BEE2 TwoSuper dataset with different sizes

| Model | Test Acc (64x64) | Val Acc (64x64) | Test Acc (128x128) | Val Acc (128x128) | Test Acc (256x256) | Val Acc (256x256) |
|---|---|---|---|---|---|---|
| RF(50) | 95.44 | 69.28 | 95.42 | 71.17 | 95.66 | 70.40 |
| RF(100) | 95.48 | 69.53 | 95.41 | 71.04 | 95.69 | 70.44 |
| RF(150) | 95.32 | 69.92 | 95.43 | 70.75 | 95.72 | 70.47 |
| SVM | 50.41 | 48.89 | 65.18 | 55.08 | 65.72 | 54.92 |

Table 6.9: Results after applying PCA on features on BEE3 dataset with different sizes

| Model | Test Acc (64x64) | Val Acc (64x64) | Test Acc (128x128) | Val Acc (128x128) |
|---|---|---|---|---|
| RF(50) | 86.81 | 75.56 | 86.95 | 76.35 |
| RF(100) | 86.76 | 75.39 | 87.13 | 76.23 |
| SVM | 51.65 | 53.97 | 45.89 | 51.22 |

## 6.4 Models

This section shows the test and validation accuracy of different models on BEE1, BEE2 and BEE3 datasets.

- **State of the art models** : We also trained and validated state of the art networks to see their performance with our datasets. The results for 6 such networks are clearly shown in tables 6.10, 6.11, 6.12 and 6.13.

- **Machine Learning** : An image of size 64x64 with 3 channels is flattened to a one dimensional array of 12288. It is normalized using a min-max normalization. These flattened images are trained with 20,40,60,80 and 100 trees using Random Forest classifier and SVM classifier using 'L2' penalty.

- **Feature Selection** : The 21 features are indexed from 0 to 20. We manually selected the number of features required and set the value to 14 in both SFS and UFS methods.

- **Other Networks** : We used 10 Convolution networks as proposed in prateeks paper. ConvNetGS-1, ConvNetGS-2, ConvNetGS-3, ConvNetGS-4 and ConvNetGS-5 are automatically designed ConvNets using greedy grid search algorithm. ConvNet

1, ConvNet 2, ConvNet 3, ConvNet 4, ConvNet 5, ConvNet 6, ConvNet 7, ConvNet 8, ConvNet 9, ConvNet 10 are other ConvNets [42]. ConvNet 11, ConvNet 12 and ConvNet 13 are the best architectures designed for BEE3 dataset. The architecture of the best model is shown in Fig.6.2.

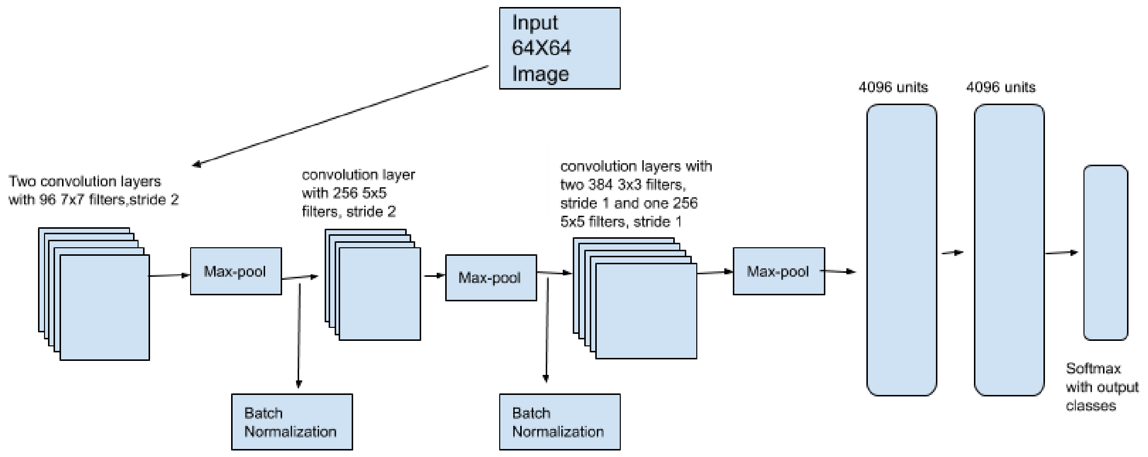
Input 64X64 Image
Two convolution layers with 96 7x7 filters,stride 2
Max-pool
convolution layer with 256 5x5 filters, stride 2
Max-pool
convolution layers with two 384 3x3 filters, stride 1 and one 256 5x5 filters, stride 1
Max-pool
4096 units
4096 units
Softmax with output classes
Batch Normalization
Batch Normalization

Fig. 6.2: Best model on BEE3 dataset

## 6.5 Performance Results

Table 6.10: Results on BEE1 dataset

| **Model** | **Test Accuracy** | **Validation Accuracy** |
|---|---|---|
| ResNet | 99.78 | 99.37 |
| ConvNet 1 | 99.85 | 99.36 |
| ConvNet 13 | 99.84 | 99.33 |
| GoogleLeNet | 99.78 | 99.20 |
| ZFNet | 99.80 | 99.17 |
| ConvNet 11 | 99.76 | 99.14 |
| ConvNetGS-5 | 99.69 | 99.10 |
| VGGNet 16 | 99.80 | 99.09 |
| ConvNetGS-4 | 99.77 | 99.08 |
| ConvNet 3 | 99.81 | 99.08 |
| ConvNetGS-3 | 99.69 | 99.02 |
| ConvNet 2 | 99.43 | 99.02 |
| ConvNet 12 | 99.58 | 98.97 |
| AlexNet | 99.63 | 98.86 |
| ConvNetGS-2 | 99.57 | 98.76 |
| ConvNet 7 | 99.48 | 98.50 |
| ConvNet 6 | 99.29 | 98.45 |
| ConvNetGS-1 | 99.25 | 97.84 |
| ConvNet 5 | 99.05 | 97.58 |
| ConvNet 10 | 99.12 | 97.31 |
| ConvNet 8 | 99.25 | 96.43 |
| SFS | 97.91 | 96.96 |
| UFS | 97.64 | 95.77 |
| ConvNet 4 | 97.85 | 95.55 |
| RF(80) | 97.52 | 93.22 |
| RF(100) | 97.54 | 93.14 |
| RF(60) | 97.41 | 93.11 |
| RF(40) | 97.40 | 92.63 |
| RF(20) | 97.06 | 92.48 |
| SVM | 55.36 | 55.18 |
| ConvNet 9 | 50 | 50 |

Table 6.11: Results on BEE2 OneSuper dataset

| **Model** | **Test Accuracy** | **Validation Accuracy** |
|---|---|---|
| VGGNet 16 | 93.82 | 91.52 |
| ConvNet 11 | 94.17 | 90.31 |
| ConvNet 2 | 93.69 | 89.42 |
| ConvNet 3 | 92.57 | 89.48 |
| ConvNet 6 | 94.21 | 88.96 |
| AlexNet | 93.69 | 88.05 |
| ConvNet 13 | 94.23 | 87.28 |
| ConvNet 12 | 94.08 | 86.71 |
| ConvNet 1 | 94.22 | 85.59 |
| ZFNet | 93.97 | 84.90 |
| GoogleLeNet | 94.12 | 84.52 |
| ConvNetGS-5 | 92.82 | 84.0 |
| ConvNetGS-4 | 93.50 | 83.95 |
| ResNet | 93.48 | 83.73 |
| ConvNetGS-1 | 91.66 | 83.69 |
| ConvNetGS-3 | 92.62 | 83.41 |
| ConvNetGS-2 | 91.74 | 82.84 |
| ConvNet 8 | 94.21 | 81.14 |
| ConvNet 7 | 93.59 | 80.85 |
| ConvNet 5 | 92.40 | 77.24 |
| ConvNet 10 | 92.81 | 72.11 |
| RF(60) | 89.88 | 61.78 |
| RF(40) | 89.86 | 61.72 |
| RF(100) | 90.06 | 61.08 |
| RF(80) | 89.64 | 60.81 |
| RF(20) | 89.35 | 59.62 |
| SFS | 89.58 | 52.75 |
| UFS | 88.71 | 46.57 |
| ConvNet 4 | 76.16 | 50 |
| ConvNet 9 | 76.16 | 50 |
| SVM | 79.65 | 33.02 |
| VGGNet 19 | 76.16 | 24.31 |

Table 6.12: Results on BEE2 TwoSuper dataset

| **Model** | **Test Accuracy** | **Validation Accuracy** |
|---|---|---|
| ResNet | 96.82 | 78.57 |
| VGGNet 16 | 98.18 | 75.50 |
| GoogleLenet | 98.50 | 75.29 |
| ZFNet | 98.60 | 74.96 |
| ConvNet 6 | 97.35 | 74.31 |
| ConvNet 13 | 98.19 | 74.19 |
| RF(100) | 97.29 | 73.77 |
| ConvNet13 | 98.57 | 73.64 |
| ConvNetGS-4 | 98.26 | 73.43 |
| RF(80) | 97.26 | 73.07 |
| ConvNet 11 | 98.32 | 73.04 |
| RF(60) | 97.29 | 72.91 |
| RF(40) | 97.24 | 72.50 |
| AlexNet | 98.39 | 72.43 |
| RF(20) | 96.96 | 72.34 |
| ConvNet 8 | 97.56 | 72.80 |
| ConvNet 2 | 97.83 | 71.90 |
| ConvNet 12 | 98.52 | 71.07 |
| ConvNetGS-5 | 98.14 | 70.41 |
| ConvNet 10 | 97.46 | 70.45 |
| ConvNetGS-3 | 97.95 | 70.26 |
| SVM | 87.93 | 69.34 |
| ConvNet 7 | 97.27 | 68.50 |
| SFS | 95.43 | 68.31 |
| ConvNet 1 | 98.45 | 68.17 |
| ConvNetGS-1 | 96.88 | 68.10 |
| ConvNetGS-2 | 97.52 | 68.02 |
| ConvNet 3 | 98.16 | 67.04 |
| ConvNet 5 | 97.44 | 66.0 |
| UFS | 95.70 | 64.09 |
| VGGNet 19 | 55.95 | 57.86 |
| ConvNet 4 | 55.95 | 50 |
| ConvNet 9 | 55.95 | 50 |

Table 6.13: Results on BEE3 dataset

| **Model** | **Test Accuracy** | **Validation Accuracy** |
|---|---|---|
| ResNet | 96.58 | 91.0 |
| ConvNet 12 | 96.54 | 91 |
| ConvNet 13 | 96.56 | 90.79 |
| ConvNet 4 | 95.57 | 90.59 |
| AlexNet | 95.93 | 90.57 |
| ConvNet 1 | 96.40 | 90.55 |
| ZFNet | 96.45 | 90.47 |
| ConvNet 11 | 96.40 | 90.09 |
| ConvNetGS-4 | 95.91 | 89.43 |
| GoogleLenet | 95.93 | 88.63 |
| ConvNet 9 | 95.46 | 88.58 |
| ConvNetGS-3 | 94.79 | 88.30 |
| ConvNetGS-5 | 95.04 | 87.89 |
| ConvNet 2 | 93.59 | 87.35 |
| ConvNet 3 | 93.03 | 87.34 |
| VGGNet 16 | 96.01 | 86.88 |
| ConvNetGS-2 | 94.19 | 86.22 |
| ConvNet 6 | 88.14 | 84.74 |
| RF(80) | 92.67 | 83.36 |
| ConvNetGS-1 | 91.16 | 83.08 |
| RF(100) | 92.67 | 83.07 |
| RF(60) | 92.23 | 82.94 |
| ConvNet 10 | 89.77 | 82.93 |
| ConvNet 7 | 88.89 | 82.67 |
| RF(40) | 92.18 | 82.55 |
| RF(20) | 91.65 | 80.78 |
| ConvNet 8 | 83.36 | 78.77 |
| SFS | 89.23 | 76.87 |
| UFS | 85.74 | 76.28 |
| SVM | 73.53 | 65.34 |
| ConvNet 5 | 54.85 | 47.74 |

### 6.5.1 Convolution Neural Network Models with different parameters on BEE3 dataset

Convolution Network architecture is developed to classify images obtained from videos. We trained the network to classify 64x64 RGB image ad bee, no-bee or shadow bee. Architecture of the models are explained below.

### 6.5.2 Deep Networks

To start with, we trained our network with 2 conv layers. The first convolutional layer has 32 4x4 filters and second convolutional layer has 64 4x4 filters. Each convolutional layer is followed by a max-pooling layer with kernel size 3 and then the output layer. We used tanh as activation function and learning rate is 0.01.

A fully connected layer is added to this network to see its impact. Table 6.14 shows the results. By adding FC layer with 128 neurons, there is 5% increase in train accuracy and 2% increase in test accuracy but no significant difference in the validation accuracy.

Table 6.14: Effect of FC layer on BEE3 dataset

| Network | Train Acc | Train Loss | Test Acc | Test Loss | Validation Accuracy |
|---|---|---|---|---|---|
| 2-conv layers | 93.67% | 0.16 | 93.99% | 0.169 | 88.01% |
| 1 FC layer | 98.11% | 0.076 | 95.44% | 0.130 | 87.13% |

To see the effect of filter size in convolutional layer, its size is increased to 5 and learning rate is decreased to 0.001. Table 6.15 has the results with two CONV layers and kernel size as 4, then changing the activation function to tanh and optimizer to ADAM. In the next network, the kernel size is changed, rest all are same as first network.

Table 6.15: Effect of kernel size and optimizer on BEE3 dataset

| Network | Train Acc | Train Loss | Test Acc | Test Loss | Validation Accuracy |
|---|---|---|---|---|---|
| k = 4 | 87.21% | 0.309 | 89.04% | 0.289 | 82.60% |
| ADAM | 97.69% | 0.054 | 95.53% | 0.159 | 88.06% |
| k = 7 | 86.65% | 0.342 | 88.74% | 0.312 | 81.53% |

From both the tables 6.14 and 6.15, it is observed that the accuracy is reduced as the learning rate value is decreased. There is no difference in the validation accuracy for the adam optimizer.

### 6.5.3 Deeper Networks

This section deals with CNNs with 3 or more convolutional layers.

#### 3 Convolution layers

Initially, we trained our network (Network 1) with 3 CONV layers and 1 FC layer. The first convolutional layer has 16 3x3 filters, second convolutional layer has 32 3x3 filters and third has 64 3x3 filters. All the convolutional layers are followed by a max-pooling layer with kernel size 5. The FC layer has 64 neurons. ReLU is used as activation function and SGD as optimizer with 0.1 as learning rate.

To see the effect of number of neurons in convolutional layers, we trained another network (Network 2) with 32 3x3 filters in first convolutional layer, 64 3x3 filters in second and third convolutional layers. Same as network 1, the convolutional layers are followed by a max-pooling layer with kernel size as 3. A network (Network 3) with minor changes to the previous network are made like filter size is set to 5, kernel size to 7 and number of neurons in FC layers are increased to 256.

Table 6.16: Effect of number of neurons and filter size on BEE3 dataset

| Network no. | Train Acc | Train Loss | Test Acc | Test Loss | Validation Accuracy |
|---|---|---|---|---|---|
| 1 | 98.77% | 0.034 | 95.17% | 0.180 | 87.47% |
| 2 | 99.29% | 0.017 | 95.57% | 0.232 | 88.14% |
| 3 | 99.72% | 0.012 | 95.97% | 0.188 | 87.96% |

#### 4 Convolution layers

Going deeper, we designed a network with 4 convolutional layers and 2 FC layers. The first and second convolutional layers has 64 filters with size 4. Next two convolution layers has 128 4x4 filters. Each convolutional layer is followed by a max-pooling layer with kernel size as 3. The hidden layers in FC layers has 256 and 128 neurons. Learning rate is 0.01 and activation function in all the layers is tanh.

Table 6.17 shows the accuracy and loss of training, testing and validation for the networks with 4 CONV layers. Trained and tested with ADAM optimizer, by adding one more convolution layer with 256 4x4 filters, FC layer with 64 neurons added to previous network. We can observe poor performance of the model when the optimizer is changed to ADAM. The train and test loss are very loss which is not a good sign to use the model. The accuracy is degraded when a FC layer is added to 5 layered convolutional network. It is not advisable to add more FC layers to this network.

Table 6.17: COVN layers with FC layers and different optimizers on BEE3 dataset

| Network | Train Acc | Train Loss | Test Acc | Test Loss | Validation Accuracy |
|---|---|---|---|---|---|
| 4 conv layers | 99.74% | 0.0098 | 96.01% | 0.141 | 88.10% |
| ADAM | 34.55% | 1.15 | 34.01% | 1.20 | 35.18% |
| 5 conv layers | 99.77% | 0.017 | 96.26% | 0.150 | 88.24% |
| dense64 | 97.53% | 0.072 | 95.29% | 0.171 | 86.25% |

We designed another 4 convolutional layered network. First convolutional layer has 32 4x4 filters, second and third convolutional layers has 64 4x4 filters and last convolutional layer has 128 4x4 filters. A max-pooling layer with 3 as kernel size is attached after every convolutional layer. FC layer with 256 and 128 neurons are applied to the network. Tanh is the activation function and SGD is optimizer. From table 6.18, we can say that by changing filter size to 5, kernel size to 4 and changing number of filters in third and fourth convolution layers to 128, the validation accuracy is reduced whereas changing the optimizer to ADAM and activation to ReLU improved the performance to 90%.

We trained an architecture with kernel size 3 and stride with 2 in all max-pool layers. The first convolution layer has 96 7x7 filters and strides with 2. Followed by max-pool and batch normalization layers. The second convolution layer consists of 356 5x5 filters with 4 strides. The output is max-pooled and batch normalized. Then comes three convolution

Table 6.18: 4 COVN layers with different kernel size and optimizers on BEE3 dataset

| Network | Train Acc | Train Loss | Test Acc | Test Loss | Validation Accuracy |
|---|---|---|---|---|---|
| k = 3 | 99.52% | 0.016 | 95.30% | 0.168 | 84.60% |
| k = 4 | 33.71% | 1.097 | 33.97% | 1.098 | 35.24% |
| ADAM & ReLU | 99.64% | 0.011 | 95.89% | 0.242 | 90.76% |

layers with 512, 1024 and 512 1x1 filters respectively with stride as 1. Max-pooling layer is attached to it and 2 FC layers with 4096 neurons each. ReLU is the activation function.

A similar network is designed with 128 filters in thrid, 256 in fourth and 128 in fifth convolution layers. Neurons in FC layers are changed to 512. Results of these architectures are shown in table 6.19. There is no significant difference in the accuracy of these models.

Table 6.19: 5 COVN layers with different filters on BEE3 dataset

| Neurons in FC layer | Train Acc | Train Loss | Test Acc | Test Loss | Validation Accuracy |
|---|---|---|---|---|---|
| 4096 | 99.00% | 0.015 | 96.29% | 0.235 | 88.54% |
| 512 | 95.68% | 0.039 | 31.95% | 0.236 | 88.59% |

## 6.6 Algorithm on videos

The input of the algorithm is a 30 seconds mp4 video collected from a hive. This video is fed to the algorithm and the motion detection algorithm checks for any motion in the video. If the motion detection algorithm encounters any motion, it draws a contour and crops the motion a region across it. The obtained image is then fed to the classifier to check whether the motion detected region is a bee, no bee or a shadow bee. If the motion detection algorithm doesn't encounter any motion in the video then the algorithm terminates once the the whole video is processed. Figure 6.3 describes the pictorial representation of detecting the bee motions in the videos.

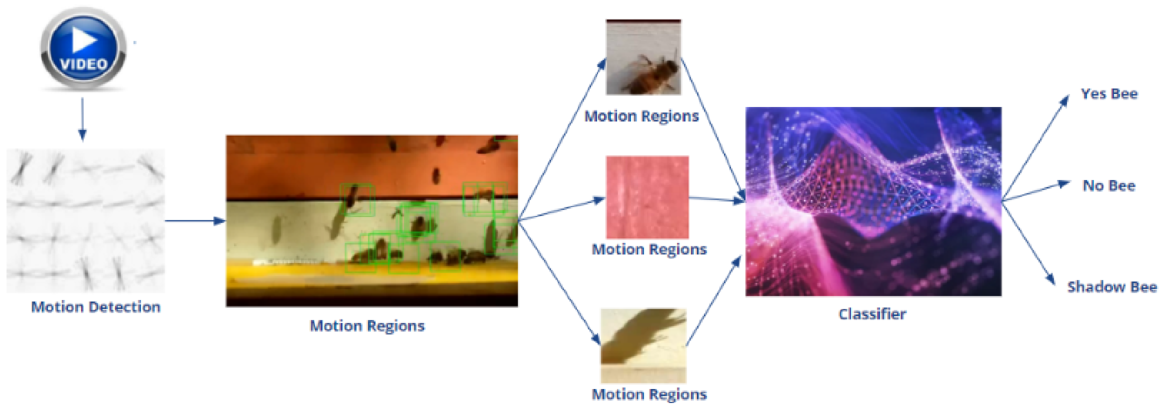

Fig. 6.3: Algorithm using ternary classification

## 6.7 Performance metrics on videos

- **Precision** - the proportion of positive identifications that are actually correct i.e. the fraction of relevant instances among the retrieved instances.

$$Precision = \frac{TP}{TP + FP} \tag{6.1}$$

- **Recall** - the proportion of actual positives that are identified correctly i.e. the fraction of relevant instances that have been retrieved over the total amount of relevant instances.
$$Recall = \frac{TP}{TP + FN} \tag{6.2}$$

- **Accuracy** - the proportion of true results among the total number of cases examined i.e. fraction of predictions model got right.
$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.3}$$

We tested the algorithm of detecting bee and shadow bee motions for 10 videos from four hives with different traffic levels by using the BEE3 classifier. The performance metrics for 10 videos are tabulated in table 6.20.

The number of bees detected correctly are True Positives(TP), number of shadow bees correctly detected are True Negatives(TN), number of bees detected as shadow bees are False Positives(FP) and number of shadow bees detected as bees are False Negatives(FN).

Table 6.20: Performance metrics of videos

| Video No. | Precision | Recall | Accuracy |
|---|---|---|---|
| 1 | 97.90 | 97.33 | 95.52 |
| 2 | 99.18 | 95.33 | 95.03 |
| 3 | 99.36 | 95.10 | 94.77 |
| 4 | 99.88 | 96.74 | 96.76 |
| 5 | 100 | 89.74 | 90.09 |
| 6 | 99.78 | 90.87 | 88.49 |
| 7 | 99.41 | 97.26 | 96.81 |
| 8 | 97.15 | 99.91 | 93.91 |
| 9 | 99.55 | 98.72 | 98.34 |
| 10 | 98.12 | 96.95 | 95.39 |

## 6.8 Comparison with actual data

We took high level, medium level and low level traffic videos and counted the number of bee motions and shadow motions. Table 6.21 shows the number of bee and shadow motions in videos counted by human and the system. We can observe that more than 85% of shadow bee motions are detected for medium and low level traffic videos. The system is unable to detect all the shadow motions for high level traffic videos. The number of bee motion counts for three videos are overestimated than actual bee motions.

Obtaining ground truth for omnidirectional bee traffic videos is very labour intensive. It took 19 hours to count number of bee and shadow motions for three videos with different traffic levels.

Table 6.21: Actual counts of videos

| Traffic level | Human bee motion counts | System bee motion counts | Human shadow motion counts | System shadow motion counts |
|---|---|---|---|---|
| High | 8863 | 12,793 | 2612 | 1139 |
| Medium | 6853 | 25,800 | 951 | 925 |
| Low | 3089 | 11,489 | 1107 | 1030 |

# CHAPTER 7
# CONCLUSION AND FUTURE WORK

This thesis examines various methods that can be used to analyze bee traffic which further contributes to monitor the health of bee hives. We evaluated the results of Machine Learning, Deep Learning and other techniques on different datasets. However, all other techniques fail to achieve the level of ConvNets. Let's go deep into our results for different datasets.

## BEE1

ResNet achieved the highest accuracy of all the networks. All other state of the art models scored above 98% of accuracy and other ConvNets also performed well except ConvNet9. Random Forest gave better results when trained and tested on features of the images than raw images. The validation accuracy is increased by 1% i.e. 97%, when the image is resized to 64x64 before extracting features from the images. We can say that feature extraction technique can help to perform on par with deep learning methods.

## BEE2

On BEE2_1S, VGGNet 16 outperformed all other networks with 91.52% accuracy. There is no significant difference in the accuracy's of Machine Learning techniques on raw images and features, even after resizing the images to 128x128 and 256x256 pixels. Their performance is less than 70%.

On BEE2_2S, the best accuracy is 78.57% which is achieved by ResNet. Random Forest on raw images with an accuracy of 73.77% gave better results than features and PCA. ConvNet4, ConvNet9 and VGGnet 19 are the bad architectures for this dataset.

**BEE3**

BEE3 consists of SHADOW BEE and BEE images. None of the analytical approaches gave an accuracy above 50%, except feature extraction technique to detect shadow bee and bee images. ResNet and ConvNet12 tops the list with 91% accuracy. Random Forest on raw images with an accuracy of 83.36% is better than some of the ConvNets and feature extraction and selection techniques. ConvNet5 network does not seem to work well for this dataset.

We experimented 10 videos taken from four hives using the algorithm. The results show that more than 85% of shadow bees are detected accurately for medium and low level traffic videos. For high level traffic videos, the motion detection algorithm is unable to detect all the motion regions in the video. Multiple motion regions are drawn and detected as bees for one bee motion region. These problems can be solved by improving the motion detection algorithm.

The reasons for false positives and false negatives could be because of data and the classifier. The number of shadow bees detected as bees may be small in number for one video. But when we need to analyze the bee traffic over a period of time, especially in the noon and evening shadow bees detected as bees may lead to false analysis of bee hive health.

In order to detect bees accurately in videos, it is crucial to have consistent and clean data. The videos collected from four hives are comprised of different backgrounds. Since the validation data is completely distinct from train/test datasets, the classifier's performance decreases. Some bees are detected as shadow bees, it can be improved by increasing the dataset for shadow bees. If the image has bee and also its shadow, the model is detecting the image as shadow bee for some of the images. This problem can be solved by enhancing the architecture of model, experimenting with different hyper parameters or by defining custom deep learning layers instead of using convolution layers. Feature extraction and selection techniques can be applied to new datasets and train with different machine learning algorithms.

## REFERENCES

[1] Albelwi S.; Mahmood A., "A framework for designing the architectures of deep convolutional neural networks," 2017.

[2] Szegedy C.; Liu W.; Jia Y.; Sermanet P.;Reed S.; Anguelov D.; Erhan D; Vanhoucke V.; Rabinovich A., "Going deeper with convolutions," *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[3] Pawara P.; Okafor E.; Surinta O.; Scohomaker L.;Wiering M., "Comparing local descriptors and bags of visual words to deep convolutional neural networks for plant recognition," 2017.

[4] Christina Herrick. "Survey on honey bee colonies". [Online]. Available: https://www.growingproduce.com/fruits/usda-surveys-honey-bee-colony-health/

[5] Glenny W.; Cavilgli I.; Daughenbaugh K F.; Radford R.; Kegley S E.; Flenniken M L., "Honey bee (apis mellifera) colony health and pathogen composition in migratory beekeeping operations involved in california almond pollination." 2017. [Online]. Available: https://doi.org/10.1371/journal.pone.0182814

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

[7] Khoury D S.; Barron A B.; Myerscough M R., "Modelling food and population dynamics in honey bee colonies," 2013.

[8] Mclellan A R., "Honeybee colony weight as an index of honey production and nectar flow: A critical evaluation," *Journal of Applied Ecology*, 1977.

[9] Kaiser L.; Decourtye A.; Devillers J.; Delegue M H P., "Behavioural methods to assess the effects of pesticides on honey bees," *Apidologie 33*, 2002.

[10] Barron A B.; Khoury D S.; Myerscough M R., "A quantitative model of honey bee colony population dynamics," 2011.

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

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

[13] Tiwari A., "A deep learning approach to recognizing bees in video analysis of bee traffic," Master's thesis, Utah State University, Logan, UT, 2018. [Online]. Available: https://digitalcommons.usu.edu/etd/7076/

[14] Murali S.; Govindan V K., "Shadow detection and removal from a single image using lab color space," *CYBERNETICS AND INFORMATION TECHNOLOGIES*, vol. 13, 2015.

[15] Singh K K.; Pal K.; Nigam M J., "Shadow detection and removal from remote sensing images using ndi and morphological operators," *International Journal of Computer Applications*, vol. 42, pp. 37–40, 2012.

[16] Mohanaiah P., Sathyanarayana P., GuruKumar L, "Image texture feature extraction using glcm approach," *International Journal of Scientific and Research Publications*, vol. 3, 2013.

[17] Zeiler M D., Fergus R., "Visualizing and understanding convolutional networks," *European Conference on Computer Vision*, 2014.

[18] Krizhevsky A.;Sutskever I.,Hinton G E., "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[19] He K.; Zhang X.; Ren S.; Sun J., "Deep residual learning for image recognition," *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[20] Ren S.; He K.; Girshick R.; Sun J., "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, 2015.

[21] Kulyukin V., Putnam M., Reka S, "Digitizing buzzing signals into a440 piano note sequences and estimating forage traffic levels from images in solar-powered, electronic beehive monitoring," 2016.

[22] Zivkovic,Z.;, "Improved adaptive gaussian mixture model for background subtraction," *Proceedings of 17th international conference on pattern recognition(ICPR)*, vol. 2, pp. 28–31, 2004.

[23] KaewTraKulPong,P.; Bowden,R., "An improved adaptive background mixture model for realtime tracking with shadow detection," *Proceedings of second european workshop on advanced video based surveillance systems(AVBS01)*, pp. 28–31, 2001.

[24] Zivkovic,Z.; van der Heijden,F., "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern recognition letters*, pp. 773–780, 2006.

[25] "Analysis of Variance(ANOVA)". [Online]. Available: https://en.wikipedia.org/wiki/Analysis_of_variance

[26] Warne R T., "A primer on multivariate analysis of variance (manova) for behavioral scientists," *Practical Assessment, Research  Evaluation*, vol. 19, 2014.

[27] "SKIMAGE". [Online]. Available: https://scikit-image.org/docs/dev/api/skimage.feature.html

[28] "OTSU Thresholding". [Online]. Available: https://scikit-image.org/docs/0.14.x/auto_examples/xx_applications/plot_thresholding.html

[29] "Gaussian Blurring". [Online]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_filtering/py_filtering.html

[30] "Canny Edge Detection". [Online]. Available: https://docs.opencv.org/3.1.0/da/d22/tutorial_py_canny.html

[31] Avinash Uppuluri. "GLCM texture features". [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/22187-glcm-texture-features

[32] "A One-Stop Shop for Principal Component Analysis". [Online]. Available: https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c

[33] LeCun Y.; Bengio Y.; Hinton G., "Deep learning," *Nature Publishing Group, a division of Macmillan Publishers Limited*, 2015.

[34] LeCun Y.; Kavukcuoglu K.; Farabet C., "Convolutional networks and applications in vision," *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 253–256, 2010.

[35] Nwankpa C.; Ijomah W.; Gachagan A; Marshall S, "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378*, 2018.

[36] Albawi S.; Mohammed T A.; Alzawi S., "Understanding of a convolutional neural network," *International Conference on Engineering and Technology*, 2017.

[37] Srivastava N.; Hinton G.; Krizhevsky A.; Sutskever I.; Salakhutdinov R., "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research 15*, 2014.

[38] Ioffe S.; Szegedy C., "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[39] Simonyan K.; Zisserman A., "Very deep convolutional networks for large-scale image recognition," 2015.

[40] "Support Vector Machine". [Online]. Available: https://en.wikipedia.org/wiki/Support-vector_machine

[41] Rifkin R.; Klautau A., "In defense of one-vs-all classification," *Journal of Machine Learning Research*, pp. 101–141, 2004.

[42] Vats P., "Design and evaluation of convolutional networks for video analysis of bee traffic," Master's thesis, Utah State University, Logan, UT, 2019. [Online]. Available: https://digitalcommons.usu.edu/etd/7514/

# APPENDIX

Architecture of ConvNet Models

## A.1 ConvNet11

Table A.1: Detailed specification of ConvNet11

| **ConvNet11 Configuration** | | |
|---|---|---|
| | Layers | Specification |
| Layer 1 | Conv-2D | filters = 64,filterSize = 4,strides = 1,activation = relu, weightDecay = 0.1,regularizer = none |
| Layer 2 | Maxpool-2D | kernelSize=4,strides = 1 |
| | Conv-2D | filters = 64,filterSize = 4,strides = 1,activation = relu, weightDecay = 0.1,regularizer = none |
| Layer 3 | Maxpool-2D | kernelSize=4,strides = 1 |
| | Conv-2D | filters = 128,filterSize = 4,strides = 1,activation = relu, weightDecay = 0.1,regularizer = none |
| Layer 4 | Maxpool-2D | kernelSize=4,strides = 1 |
| Layer 5 | FC | number of units = 64, activation = relu |
| Layer 6 | FC | number of units = 3, activation = softmax |
| Regression | | Optimizer = sgd, Learning Rate = 0.1, Loss = Categorical Crossentropy |

## A.2 ConvNet12 (Best Model)

Table A.2: Detailed specification of ConvNet12

| **ConvNet12 Configuration** | | |
|---|---|---|
| | Layers | Specification |
| Layer 1 | Conv-2D | filters = 96,filterSize = 7,strides = 2,activation = relu, weightDecay = 0.1,regularizer = none |
| Layer 2 | Maxpool-2D | kernelSize=3,strides = 2 |
| | Batch Normalization | |
| | Conv-2D | filters = 256,filterSize = 5,strides = 2,activation = relu, weightDecay = 0.01,regularizer = none |
| Layer 3 | Maxpool-2D | kernelSize=3,strides = 2 |
| | Batch Normalization | |
| | Conv-2D | filters = 384,filterSize = 3,strides = 1,activation = relu, weightDecay = 0.01,regularizer = none |
| Layer 4 | Conv-2D | filters = 384,filterSize = 3,strides = 1,activation = relu, weightDecay = 0.01,regularizer = none |
| Layer 5 | Conv-2D | filters = 256,filterSize = 5,strides = 1,activation = relu, weightDecay = 0.01,regularizer = none |
| Layer 6 | Maxpool-2D | kernelSize=3,strides = 2 |
| Layer 7 | FC | number of units = 4096, activation = relu |
| Layer 8 | FC | number of units = 4096, activation = relu |
| Layer 9 | FC | number of units = 3, activation = softmax |
| Regeression | | Optimizer = sgd, Learning Rate = 0.01, Loss = Categorical Crossentropy |

## A.3 ConvNet13

Table A.3: Detailed specification of ConvNet13

| **ConvNet13 Configuration** | | |
|---|---|---|
| | Layers | Specification |
| Layer 1 | Conv-2D | filters = 64,filterSize = 4,strides = 1,activation = relu, weightDecay = 0.01,regularizer = none |
| Layer 2 | Conv-2D | filters = 64,filterSize = 4,strides = 1,activation = relu, weightDecay = 0.01,regularizer = none |
| Layer 3 | Maxpool-2D | kernelSize=5,strides = 2 |
| | Conv-2D | filters = 128,filterSize = 4,strides = 1,activation = relu, weightDecay = 0.01,regularizer = none |
| Layer 4 | Conv-2D | filters = 128,filterSize = 4,strides = 1,activation = relu, weightDecay = 0.01,regularizer = none |
| Layer 5 | Maxpool-2D | kernelSize=5,strides = 2 |
| | Conv-2D | filters = 256,filterSize = 4,strides = 1,activation = relu, weightDecay = 0.01,regularizer = none |
| Layer 6 | Conv-2D | filters = 256,filterSize = 4,strides = 1,activation = relu, weightDecay = 0.01,regularizer = none |
| Layer 7 | Conv-2D | filters = 256,filterSize = 4,strides = 1,activation = relu, weightDecay = 0.01,regularizer = none |
| Layer 8 | Maxpool-2D | kernelSize=5,strides = 2 |
| | Conv-2D | filters = 512,filterSize = 4,strides = 1,activation = relu, weightDecay = 0.01,regularizer = none |
| Layer 9 | Conv-2D | filters = 512,filterSize = 4,strides = 1,activation = relu, weightDecay = 0.01,regularizer = none |
| Layer 10 | Conv-2D | filters = 512,filterSize = 4,strides = 1,activation = relu, weightDecay = 0.01,regularizer = none |
| Layer 11 | Maxpool-2D | kernelSize=5,strides = 2 |
| | Conv-2D | filters = 512,filterSize = 4,strides = 1,activation = relu, weightDecay = 0.01,regularizer = none |
| Layer 12 | Conv-2D | filters = 512,filterSize = 4,strides = 1,activation = relu, weightDecay = 0.01,regularizer = none |
| Layer 13 | Conv-2D | filters = 512,filterSize = 4,strides = 1,activation = relu, weightDecay = 0.01,regularizer = none |
| Layer 14 | Maxpool-2D | kernelSize=5,strides = 2 |
| Layer 15 | FC | number of units = 256, activation = relu |
| Layer 16 | FC | number of units = 128, activation = relu |
| Layer 17 | FC | number of units = 3, activation = softmax |
| Regression | | Optimizer = sgd, Learning Rate = 0.01, Loss = Categorical Crossentropy |