Honey Bee Hive Health Detection

Sameer Raj | V S Subhang 1801CS42 | 1801CS59

Task

- Predicting the health of the bee hive.
- Implementing the model as a service, where images taken using a mobile phone from the hive are sent to a server for analytics and the results the sent back.

Problem Definition

Every third bite of food relies on pollination by bees.

In winters, honeybee hive losses exceeds 60% in some places. How can we address this issue?

How can we better understand our bees? And most importantly, how can we save them before it's too late?

While many indications of hive strength and health are visible on the inside of the hive, frequent check-ups on the hive are time-consuming and disruptive to the bees' and hive in general.

By investigating the bees that leave the hive, we can gain a more complete understanding of the hive itself.

Thus, with this task, we pave the way for more intelligent hive monitoring or beekeeping in general.

Solution

- Lot of information can be extracted from bee image of particular bee hive.
- We can process and analyse this data to predict the health of the bees and thus bee hive.
- For ex: an unhealthy hive infected with varroa mites will have bees with deformed wings or mites on their backs.
- A deep learning Convolutional neural network (CNN) model can be used for this instance.

We futher see implementation details using Python, Keras and TensorFlow backend.

Implementation overview

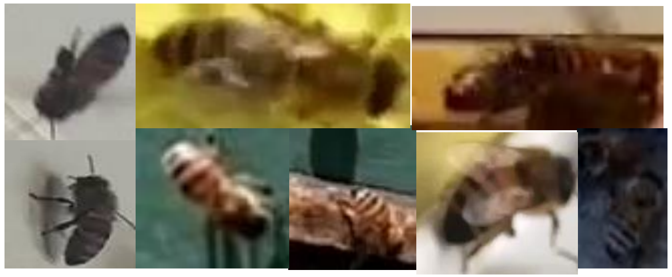
Data Set:

The dataset has been taken from Kaggle:
https://www.kaggle.com/jenny18/honey-bee-annotated-images

The dataset contains over 5000+ images of bees

The bees health has been classified into:

- Healthy
- small hive beetles
- missing queen
- hive being robbed
- ant problems
- hive beetles

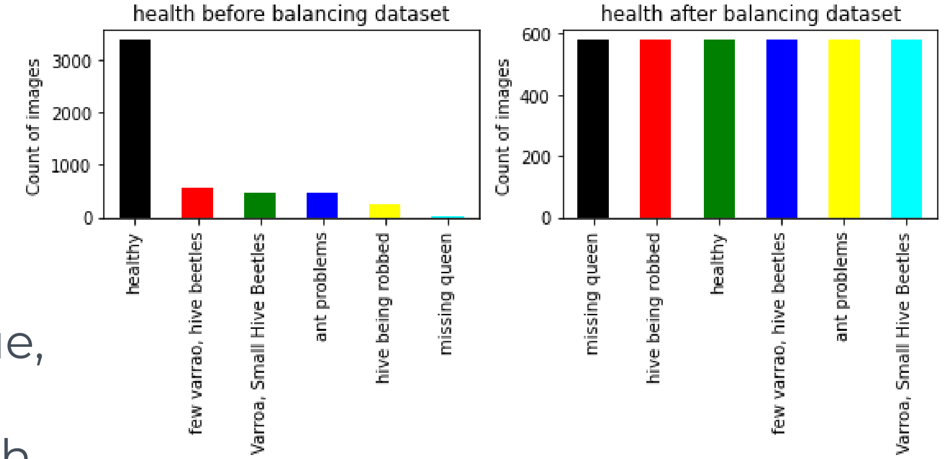


Few images from the dataset.

Implementation overview

Dataset cleaning and balancing

- We cleanse the dataset by removing bee details with missing value or bees which have no image file.
- Further, the data set is balanced. For each health issue, the data is not equal, thus it is balanced and equated for each health to ensure no biasness in prediction.

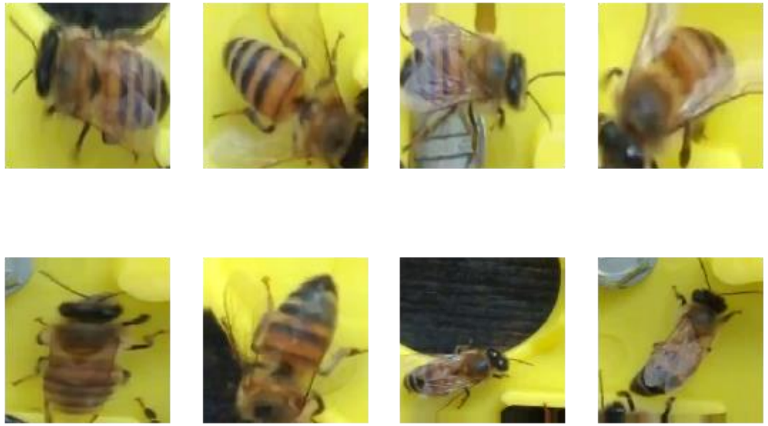


Implementation overview

Data Augmentation

Data augmentation is a techniques used to increase the amount of data by adding slightly modified copies of already existing data.

This is done through ImageDataGenerator from Keras. It includes various operations like rotating image by some angle, or flipping etc.

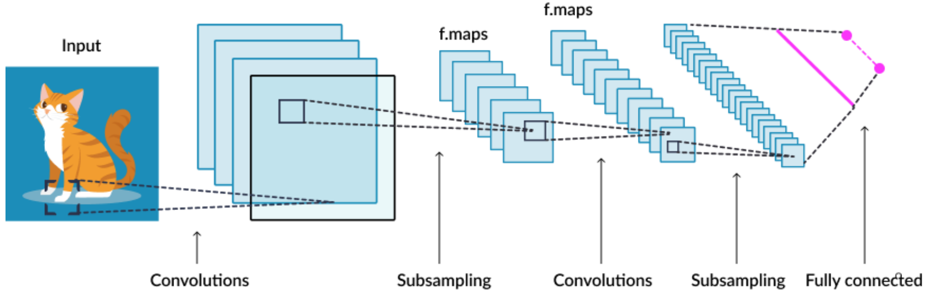


Sample augmentation results

Implementation overview

Neural Network Model

- The model is a convolutional neural network based which is class of deep neural network, most commonly used to analyse visual imagery.
- The input image in this process is converted to fully connected flattened layer which retains most of the feature of the input image.
- Further, weights are calculated and result is predicted.



Implementation overview

Neural Network Model

- The intermediate steps (as shown in picture above) helps to extract features from the model.
- This is sequential type model which is linear stack of layers.
- Various kernel extracts the major feature from the image and then activation function "relu" decreases the linearity of image.
- The output of above steps is pooled. The pooling layer summarises the features present in a region of the feature map generated by a convolution layer.
- Finally the layer is flattened and modelled to fully connected layer and is further trained.

Implementation overview

Neural Network Model

The model was trained on different set of hyper parameters and the following config gave the best accuracy:

- Two intermediate convolutional layers.
- 32 kernels of dimension 3 x 3.
- ‘relu’ activation function during convolution to remove linearity
- ‘softmax’ activation function during neural network training.
- Batch size 128

The accuracy post this step was 89.6%

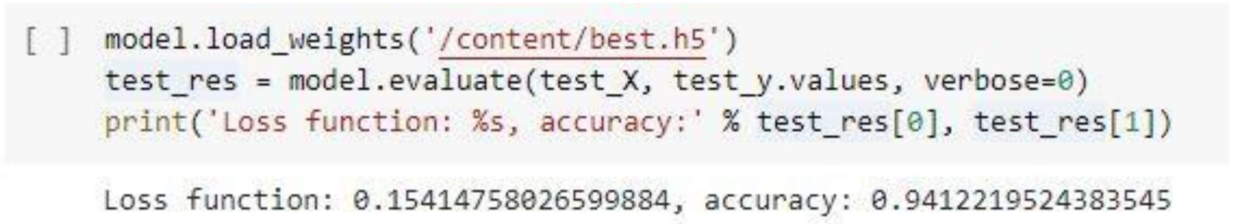
Implementation overview

Neural Network Model

The model still can be improved. Over fitting can be avoided by adding drop outs.

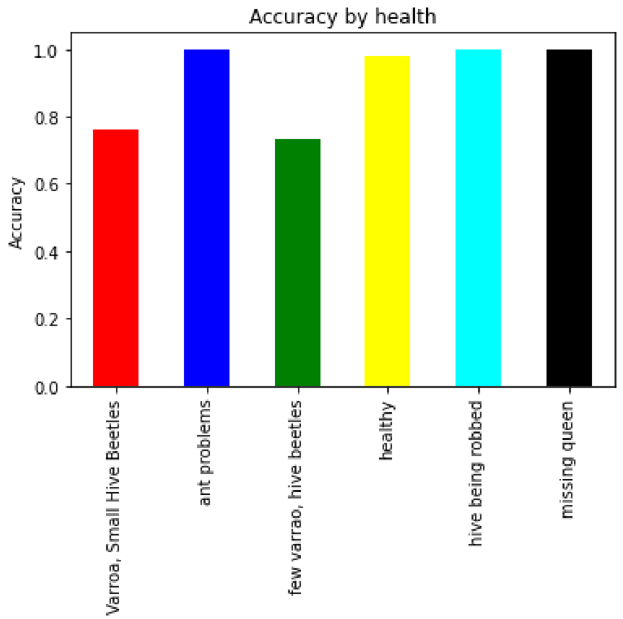
Epochs and batch size were further changed.

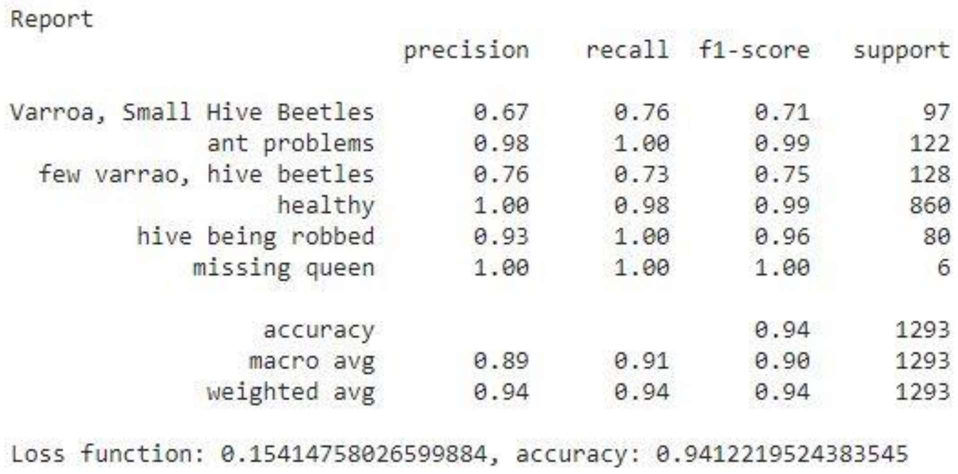
This resulted in increase of accuracy from **89.6** % to **94.1** %



```
[ ]  model.load_weights('/content/best.h5')
     test_res = model.evaluate(test_X, test_y.values, verbose=0)
     print('Loss function: %s, accuracy:' % test_res[0], test_res[1])

     Loss function: 0.15414758026599884, accuracy: 0.9412219524383545
```

Results



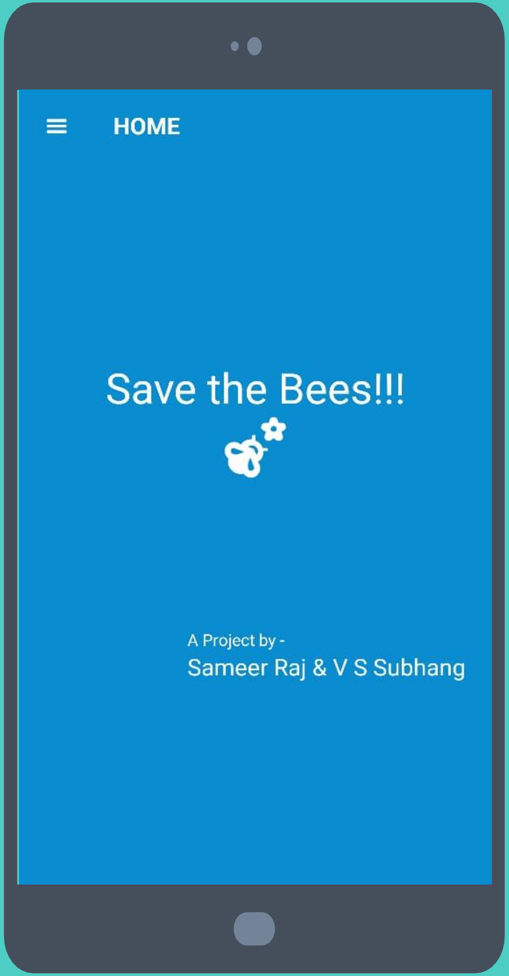

```
Report
                            precision    recall  f1-score   support

Varroa, Small Hive Beetles       0.67      0.76      0.71        97
              ant problems       0.98      1.00      0.99       122
  few varrao, hive beetles       0.76      0.73      0.75       128
                   healthy       1.00      0.98      0.99       860
         hive being robbed       0.93      1.00      0.96        80
             missing queen       1.00      1.00      1.00         6

                  accuracy                           0.94      1293
                 macro avg       0.89      0.91      0.90      1293
              weighted avg       0.94      0.94      0.94      1293

Loss function: 0.15414758026599884, accuracy: 0.9412219524383545
```

Recall = Given a class, will the classifier be able to detect it?
Precision = Given a class prediction from a classifier, how likely is it to be correct?
F1 Score = The harmonic mean of the recall and precision. Essentially, it punishes extreme values.

Implementing as a service

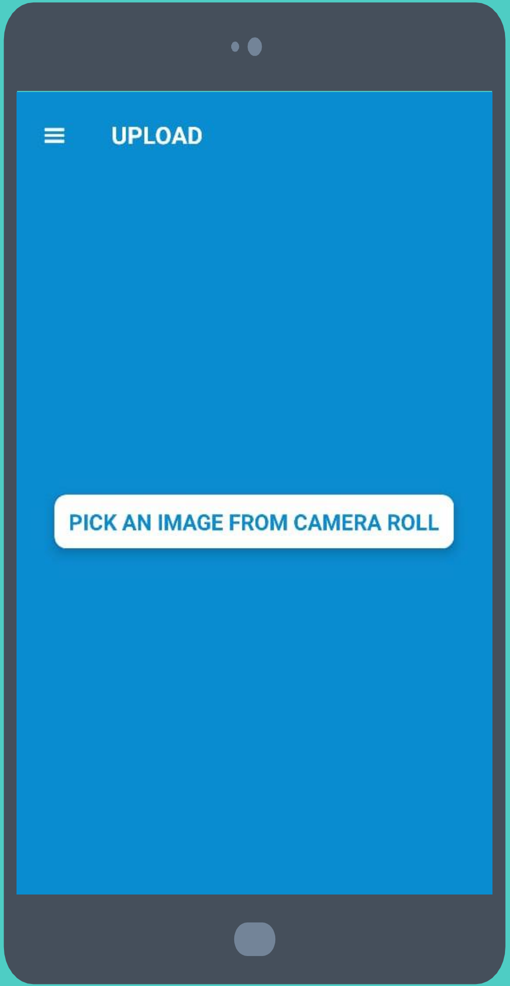
- The prediction model was deployed as a web app and Android/iOS app.
- The prediction is based on running a script on a server hosted on **Google Cloud Platform**.
- The service IP address is: http://104.197.8.34/
- The server is primarily made using **nodeJS** + **express.**
- The app is created using **react-native** and **expo** to make predictions as stated in the task.

App Screenshots

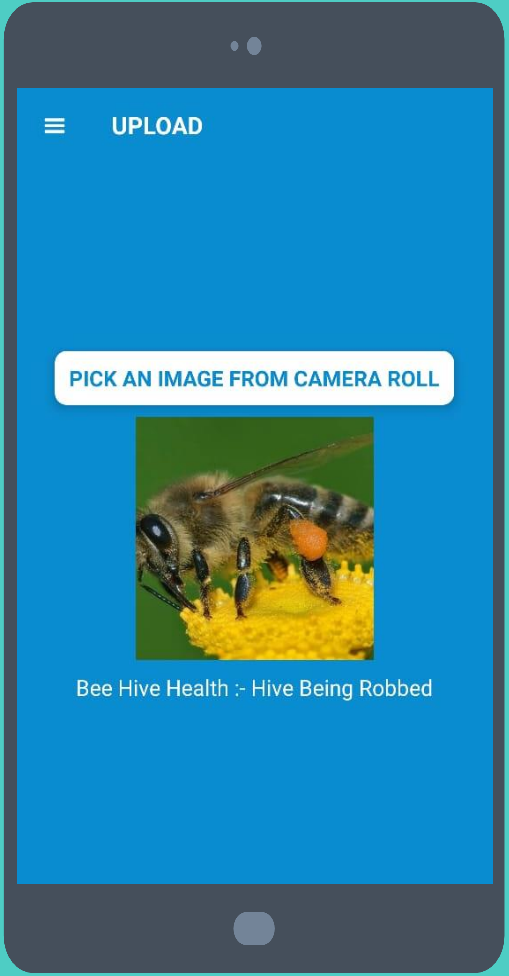

HOME

Save the Bees!!!

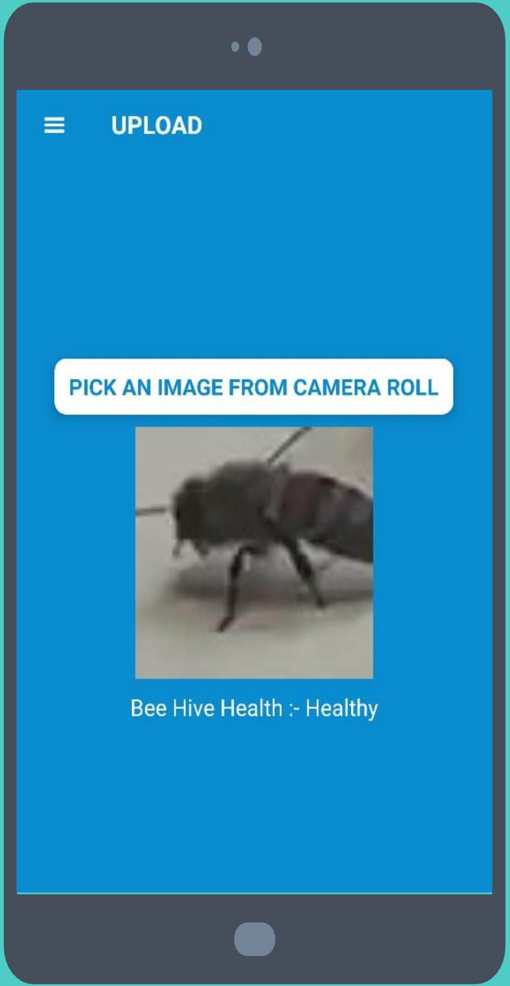
A Project by -
Sameer Raj & V S Subhang



App Screenshots



Bee Hive Health :- Hive Being Robbed



Bee Hive Health :- Healthy

Innovation

- A step towards solving real life problems and protecting nature.
- Increased accuracy by closely monitoring and brute forcing results on various hyper parameters and choosing the optimal one .
- Implementing it as ‘one of its kind’ service in form of both web app and mobile app (Android and iOS).

Challenges and learnings

Challenges

- Optimization of model was the most difficult task. We brute-forced the hyper parameters to select the optimal one.
- Selecting optimal server for deploying the model and ensuring server runs even after closing the console.

Learnings

- Various concepts like gradient descent, regression, etc and experience of working with python libraries.
- Work experience on Google Cloud Platform to deploy model.
- Work collaboration in team project.

Thank You!

Any questions?