Machine Learning Engineer Nanodegree

Capstone Project

Madhankumar B
(madhankumarbk@gmail.com)
January 3rd, 2019

I. Definition

Project Overview

Every third bite of food relies on pollination by bees. At the same time the honey bee hive losses are also increasing day by day. So we must take a steps to protect the honey bees. But, by looking at the honey bees then distinguishing its subspecies and take a steps to protect a particular species is time consuming. In order to make at least a one step from this easier machine learning models comes into play in order to differentiate species of honey bees by looking at the images. Deep learning is the subfield of machine learning which is widely used to classify image to recognize the object's in the images based on the large set of images by which it is trained on. Deep learning is one of the most promising field in machine learning where already much development has taken place and is currently used in real world application.

This project is about applying deep learning techniques to create an ML mode in order to classify the subspecies of honey bee images more accurately in this way we can easily identify the subspecies of honey bee by means of picture or image then we can take a necessary step to protect a particular honey bee. Also in this project we can explore some of the statistics about the feature of the honey bee which is going to be useful to know more about the features of the honey bee like where the particular species can be mostly found, whether it's becoming extinct or not, whether most of the subspecies of a particular kind would be healthy or not etc.

That information above mentioned can be processed from the dataset which has all the information about the features of honey bees with its picture which is taken from the various location of USA which is available publically. The deep learning is possibly applied to a huge number of images in order to classify or recognize images correctly (i.e.) deep learning learns from the experiences which indicates that more images in each subspecies will make more experience to the model.

This project is an inspiration taken from how an image of honey bee can be used to expedite the hive process, **how well the images can be used to recognize the species of honey bees** and also how can we improve hive process through the images of honey bee. The main part of this project is highlighted above.

Problem Statement

Nowadays, the honey bee hives are becoming expedite. But, pollination by bees are more important for food production so in order to save the honey bees government has taken an initiative to save honey bees from extinction, in order to do that government has to find the sub species of the particular honey bee from which it belongs which pave the way for saving honey bee.so the government has asked the machine learning engineers to develop an algorithm which accurately predicts the subspecies of honey bee given its image or picture.

Metrics

The evaluation metrics proposed are appropriate given the context of the data, the problem statement, and the intended solution. The performance of each classification model is evaluated using three statistical measures; classification accuracy, sensitivity and specificity. It is using true positive (TP), true negative (TN), false positive (FP)and false negative (FN). The percentage of Correct/Incorrect classification is the difference between the actual and predicted values of variables. True Positive (TP) is the number of correct predictions that an instance is true, or in other words; it is occurring when the positive prediction of the classifier coincided with a positive prediction of target attribute. True Negative (TN) is presenting a number of correct predictions that an instance is false, (i.e.) it occurs when both the classifier, and the target attribute suggests the absence of a positive prediction. The False Positive (FP) is the number of incorrect predictions that an instance is true. Finally, False Negative (FN) is the number of incorrect predictions that an instance is false. Table below shows the confusion matrix for a two-class classifier.

	Predicted No	**Predicted Yes**
Actual No	TN	FN
Actual Yes	FP	TP

TN – True Negative
TP – True Positive
FN – False Negative
FP – False Positive

Classification accuracy is defined as the ratio of the number of correctly classified cases and is equal to the sum of TP and TN divided by the total number of cases (TN + FN + TP + FP).

$$\text{Accuracy} = (TP + TN)/((TN + FN + TP + FP)$$

Precision is defined as the number of true positives (TP) over the number of true positives plus the number of false positives (FP).

$$\text{Precision} = TP/TP+FP$$

Recall is defined as the number of true positives (TP) over the number of true positives plus the number of false negatives (FN).

$$\text{Recall} = TP/TP+FN$$

Sensitivity refers to the rate of correctly classified positive and is equal to TP divided by the sum of TP and FN.
Sensitivity may be referred as a True Positive Rate.

Specificity refers to the rate of correctly classified negative and is equal to the ratio of TN to the sum of TN and FP

F1-score is a metric which is calculated based on the precision and recall because for some problem the result may be biased towards a precision or recall in those cases f1-score comes in handy.

$$\text{F1-score}=2*((\text{precision}*\text{recall})/(\text{precision}+\text{recall}))$$

II. Analysis

Data Exploration

The dataset "Annotated Bee Images" is a public data available in kaggle in order to solve the above problem. The data consists of images and csv files where the features about each images is present. With the help of these data's we are able to identify many features which are listed as below.

The dataset contains 2 folders with information necessary to make a prediction.
They are: 1. bee_datas.csv 2.bee_imgs.zip

Bee_datas.csv contains all the information about the bee images the details are as follows,
1.**file** - File name in bee_imgs folder.
2.**Date** - Date of video captures.
3.**time** - Time of day of video capture (military time).
4.**location** - Location (city, state, country).

5.**zip code** - Zip Code to numerically describe location.
6.**subspecies** - Subspecies of Apis mellifera species.
7.**health** - Health of a bee.
8.**pollen_carrying** - Presence of pollen on the bee's legs.
9.**caste** - Worker, Drone, or Queen bee.

bee_imgs.zip contains 5172 images of honey bee required for training.

Input variables:
The input for this problem will be an image with the corresponding filename in the csv file.The sample Input images are shown in the Fig.1

Output variable (desired target):
y – It is a subspecies from the csv file which categorically encoded with 7 classes. The distribution for this variable is shown in Fig.2.

Classes	Number of Images
-1	428
1 Mixed local stock 2	472
Carniolan honey bee	501
Italian honey bee	3008
Russian honey bee	527
VSH Italian honey bee	199
Western honey bee	37

Here the -1 represent for those images whose subspecies we didn't know.
Based on the above details Italian honey bee every species comes around same range so it will not create much problem to the balance of the dataset so as of now I am considering this as a balanced dataset if it impacts result of our final model then we will change the performance metric.

If you see the below distribution of images with respect to height and width of each image,the averages values for the images sizes are between 50 and 100 pixels and the extreme values are from few pixels to approximately 500.

The dataset is publically available via kaggle and accessible.
Data source: The Bee Image Dataset: Annotated Honey Bee Images
Apis mellifera with location, date, health, and more labels

Reference: https://www.kaggle.com/jenny18/honey-bee-annotated-images

Exploratory Visualization

Fig. 1 Sample Input Images

Fig. 2 Distribution of subspecies

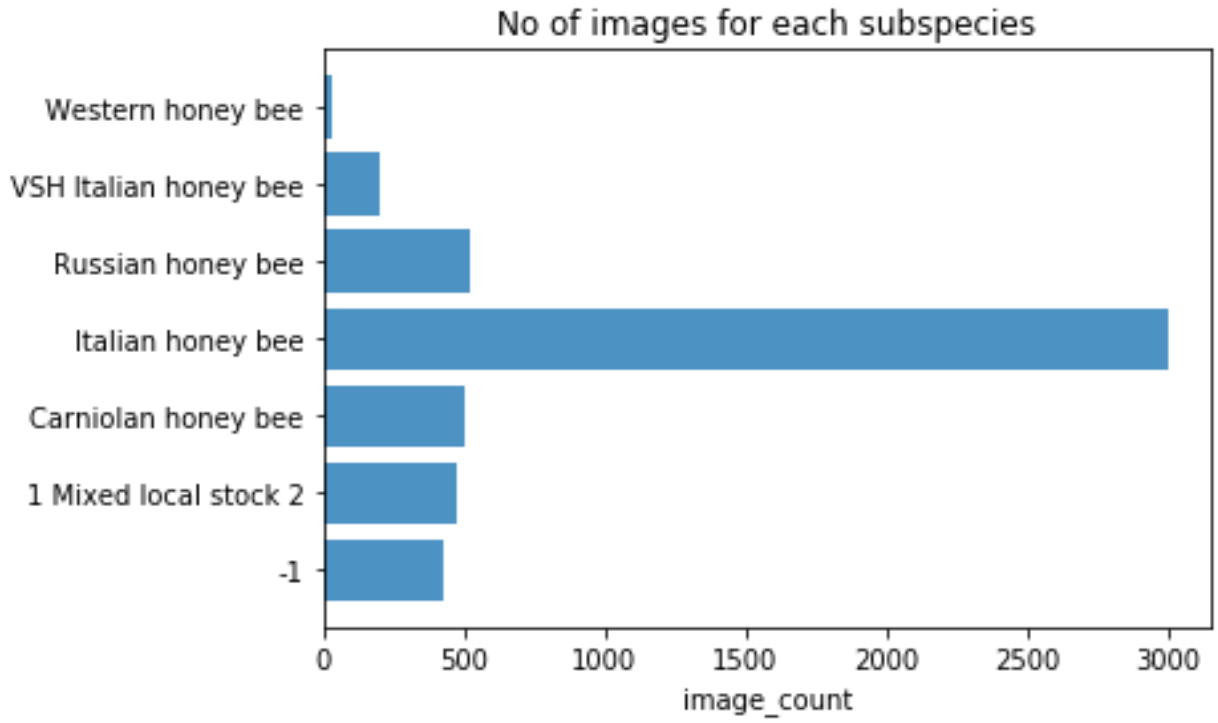
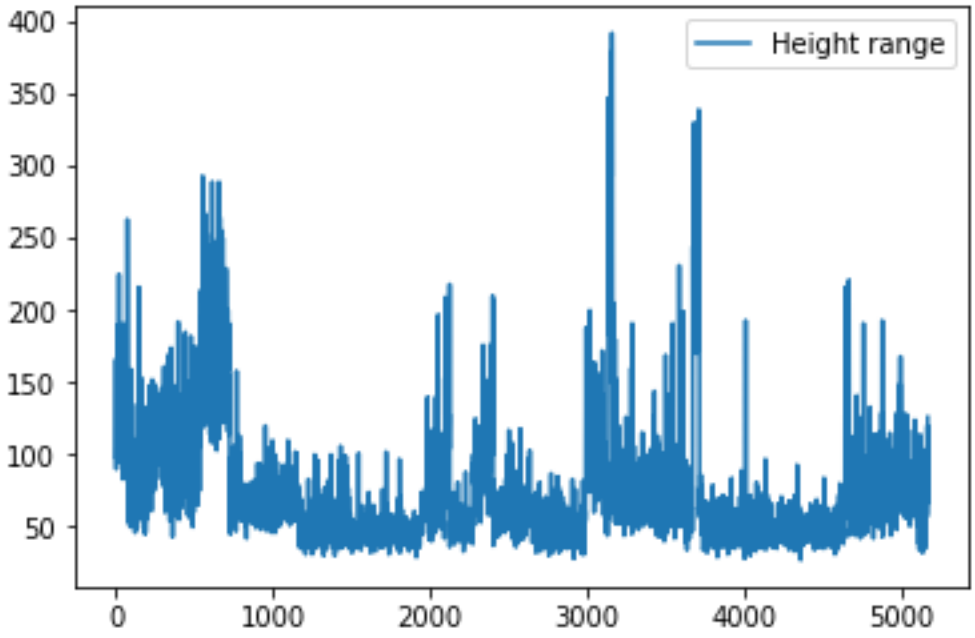


Fig. 3 Distribution of image width and height

In order to understand the images better and o preprocess the inputs before building a model we need to understand the size of the images much better because deep learning model accepts the same sized input to train the model. Let us see the distribution of each image's height and width.


Height range
400
350
300
250
200
150
100
50
0
1000
2000
3000
4000
5000

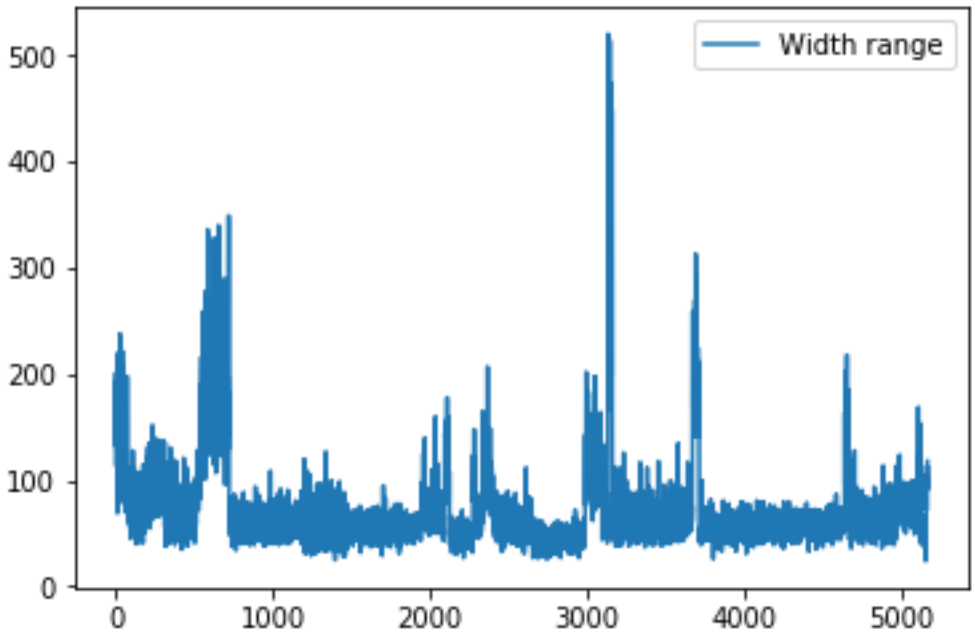

Width range
500
400
300
200
100
0
0
1000
2000
3000
4000
5000

Fig. 4 Distribution of Location.

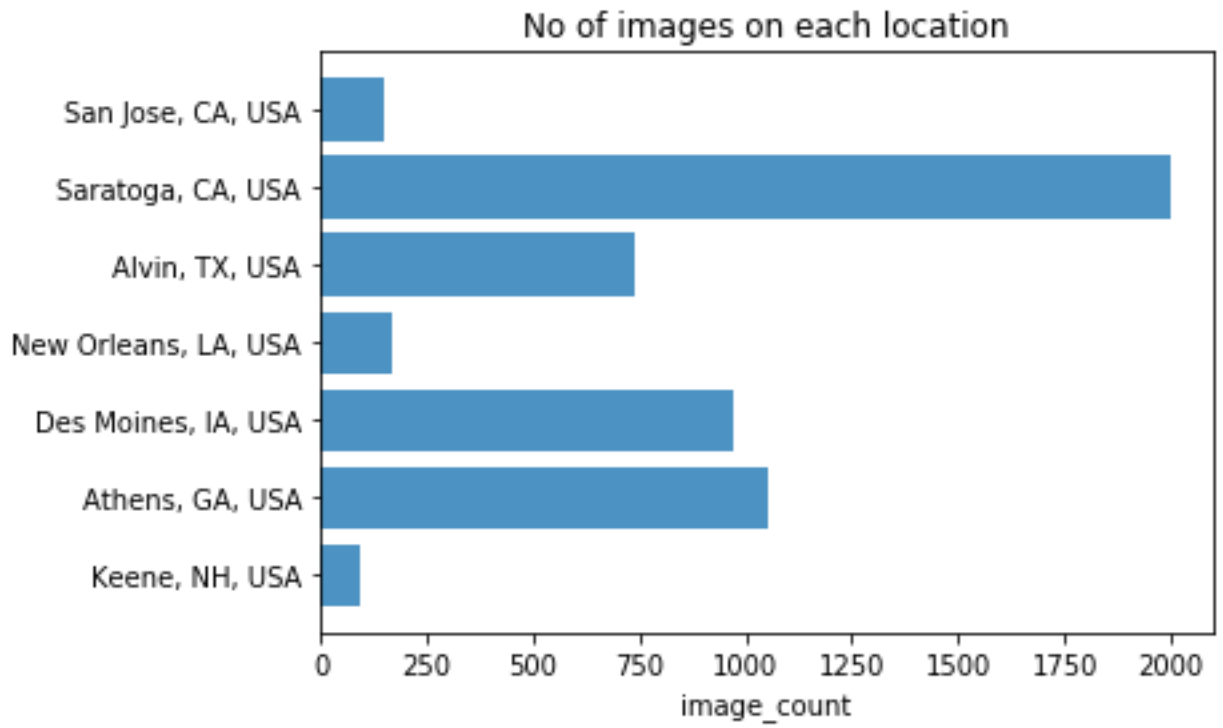
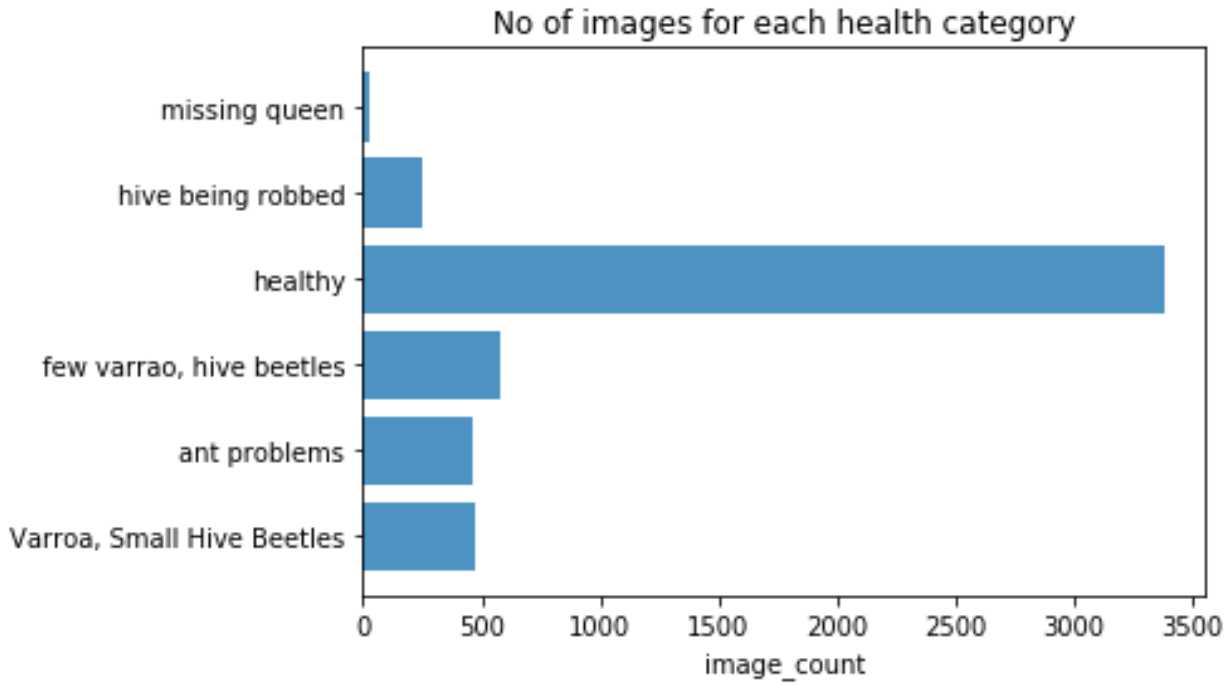


Fig.5 Distribution of Health of each species

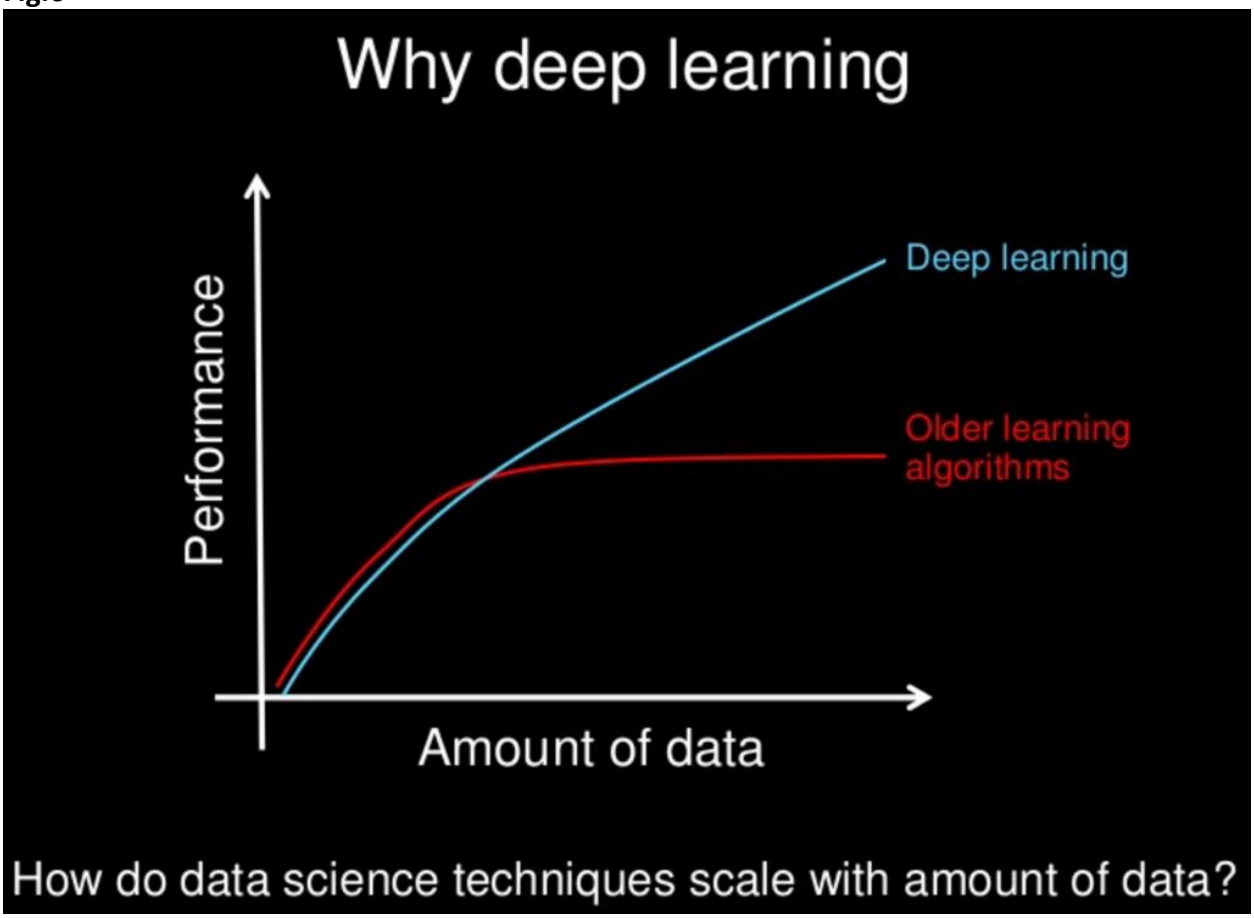


Algorithms and Techniques

There are several classification algorithms available in supervised learning but with the image as input the optimized algorithm needs to capture each and every pattern accurately in order to classify images as different classes and to recognize its species based on the images so there comes a one of the most promising field of machine learning **DEEP LEARNING.**

The Deep Learning is built based on the neural networks which means that deep learning is nothing but large or deep neural network where each and every pattern in the data is captured and manipulated.so it makes a classification more accurate compare to other classification problems with large amount of data. The relationship between amount of data and performace is shown below in fig.6

Fig.6



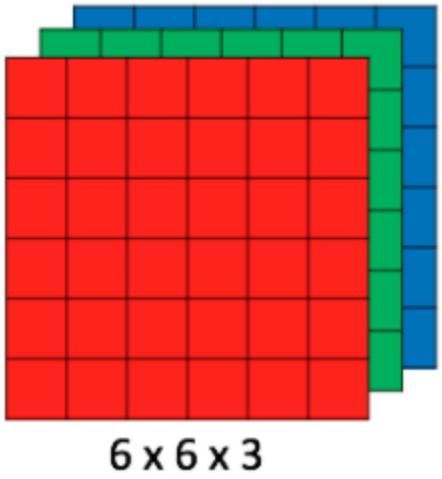
Convolution Neural Network(CNN)

There are several algorithms available in deep learning which can be categorized based on the supervised and unsupervised learning techniques.

In neural networks, Convolutional neural network (ConvNets or CNNs) is one of the main categories to do images recognition, images classifications. Objects detections, recognition faces etc., are some of the areas where CNNs are widely used.

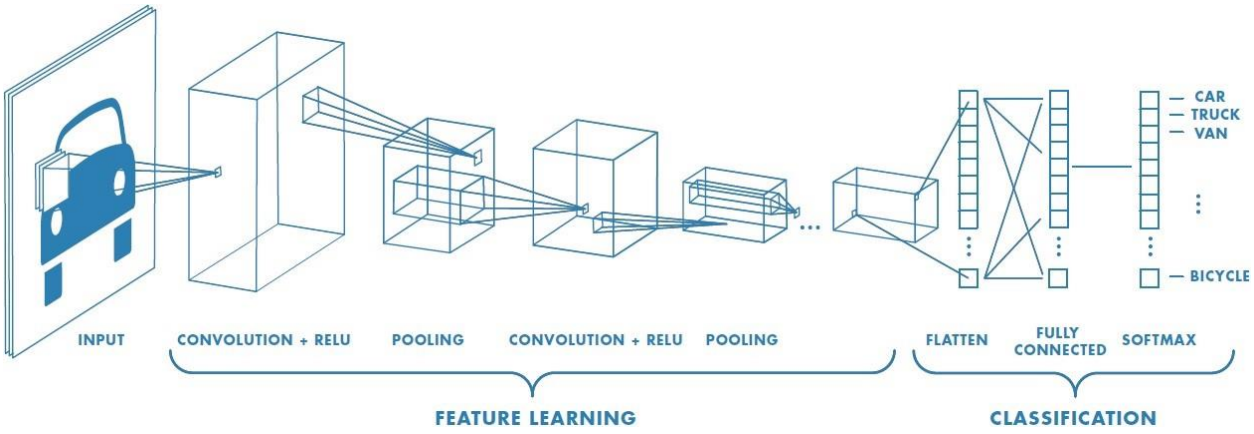
CNN image classifications takes an input image, process it and classify it under certain categories (Eg., Dog, Cat, Tiger, Lion). Computers sees an input image as array of pixels and it depends on the image resolution. Based on the image resolution, it will see h x w x d (h = Height, w = Width, d = Dimension). Eg., An image of 6 x 6 x 3 array of matrix of RGB (3 refers to RGB values) and an image of 4 x 4 x 1 array of matrix of grayscale image.

Fig.7 Array of RGB matrix



Technically, deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernels), Pooling, fully connected layers (FC) and apply Softmax function to classify an object with probabilistic values between 0 and 1. The below figure is a complete flow of CNN to process an input image and classifies the objects based on values.

Fig.8 Neural network with many convolutional layers

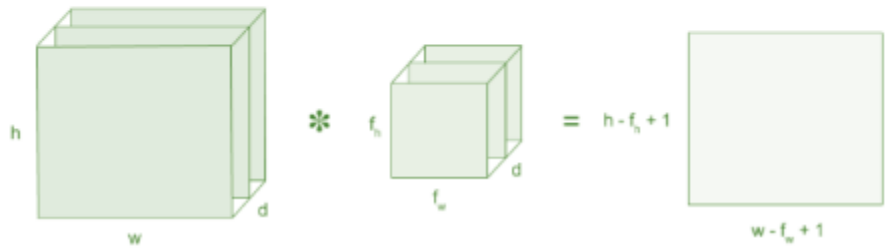


Convolution Layer

Convolution is the first layer to extract features from an input image. Convolution preserves the

relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel

- An image matrix (volume) of dimension **(h x w x d)**
- A filter **(f_h x f_w x d)**
- Outputs a volume dimension **(h - f_h + 1) x (w - f_w + 1) x 1**



Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by applying filters.

Strides

Stride is the number of pixels shifts over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and so on. The below figure shows convolution would work with a stride of 2.

Padding

Sometimes filter does not fit perfectly fit the input image. We have two options:

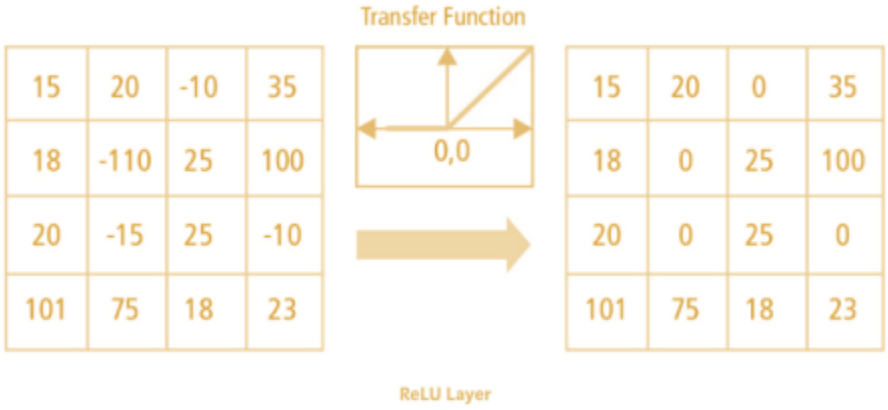
- Pad the picture with zeros (zero-padding) so that it fits
- Drop the part of the image where the filter did not fit. This is called valid padding which keeps only valid part of the image.

Non Linearity (ReLU)

ReLU stands for Rectified Linear Unit for a non-linear operation. The output is ƒ(x) = max (0, x).

Why ReLU is important: ReLU' s purpose is to introduce non-linearity in our ConvNet. Since, the real world data would want our ConvNet to learn would be non-negative linear values.

Fig.9 ReLU operation



There are other nonlinear functions such as tanh or sigmoid can also be used instead of ReLU. Most of the data scientists uses ReLU since performance wise ReLU is better than other two.

Pooling Layer

Pooling layers' section would reduce the number of parameters when the images are too large. Spatial pooling also called subsampling or down sampling which reduces the dimensionality of each map but retains the important information. Spatial pooling can be of different types:

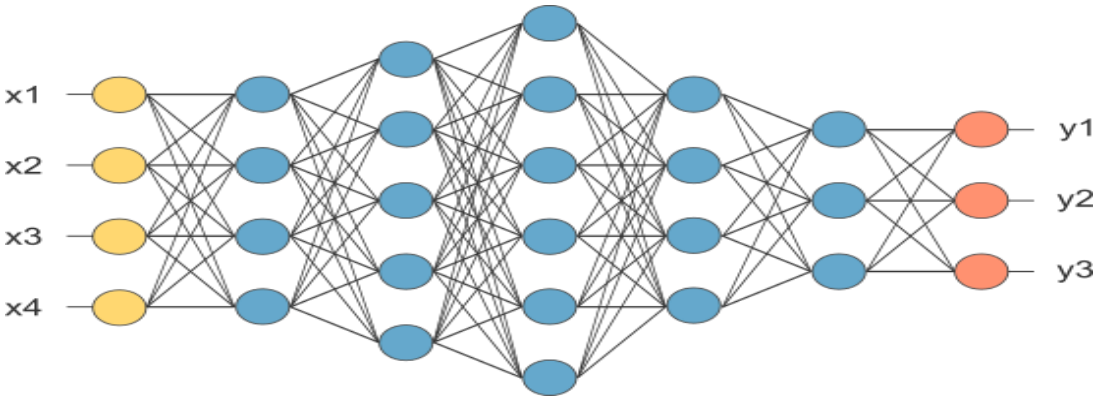
- Max Pooling
- Average Pooling
- Sum Pooling

  Max pooling takes the largest element from the rectified feature map. Taking the largest element could also take the average pooling. Sum of all elements in the feature map call as sum pooling.

Fully Connected Layer

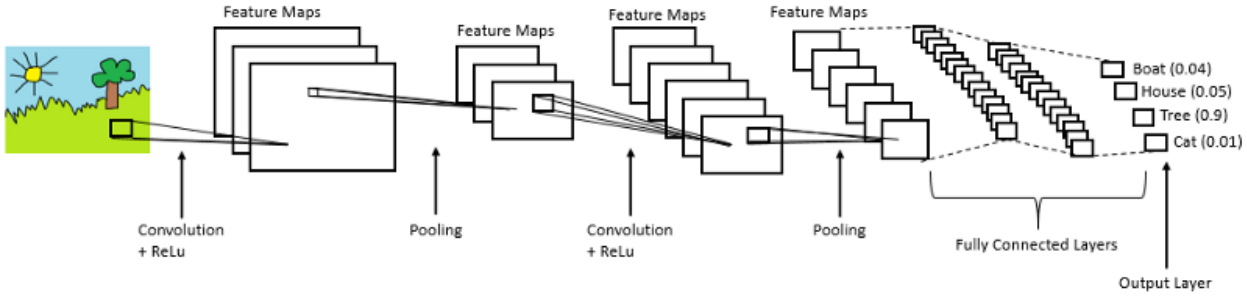
The layer we call as FC layer, we flattened our matrix into vector and feed it into a fully connected layer like neural network.

Fig.10 After pooling layer, flattened as FC layer



In the above diagram, feature map matrix will be converted as vector (x1, x2, x3, …). With the fully connected layers, we combined these features together to create a model. Finally, we have an activation function such as softmax or sigmoid to classify the outputs as cat, dog, car, truck etc.,

Fig.11 Complete CNN architecture



Benchmark Model

For this model, the benchmark model will be a Logistic regression model to classify different classes or a basic CNN model with one layer to classify the subspecies with its precision, recall and f1 score. I am going to use Multi-layer CNN model with its improvement techniques to give maximize the precision, accuracy and f1 score for this classification task.

We have tried a basic cnn model without any optimization it yields approximately 0.8550 accuracy on test set and around 82% accuracy on validation set. This simple model is implemented in the Notebook. The table below shows the metrics of benchmark model for each subspecies.

Subspecies	Precision	Recall	F1-score

-1	0.77	0.73	0.75
1 Mixed local stock 2	0.55	0.59	0.57
Carniolan honey bee	0.84	0.97	0.95
Italian honey bee	0.90	0.91	0.90
Russian honey bee	0.95	0.92	0.93
VSH Italian honey bee	0.77	0.84	0.80
Western honey bee	1.00	1.00	1.00

Loss function: 0.39376443068762335, accuracy: 0.8685990337588361
The objective of model refinement is to beat this score and implement an optimized model.

III. Methodology

Data Preprocessing

We will prepare the data by splitting feature and target/label columns and also check for quality of given data and perform data cleaning. To check if the model I created is any good, I will split the data into `training', 'validation` and `test` sets to check the accuracy of the best model. We will split the given `training` data in two ,80% of which will be used to train our models and 20% we will hold back as a `validation` set. We also have 20% of test set as a separate set in order to check the final accuracy of a trained model.

Then we will categorize each subspecies as separate classes and label them in order to differentiate each subspecies and its features. This process is known as one hot encoding in machine learning language.

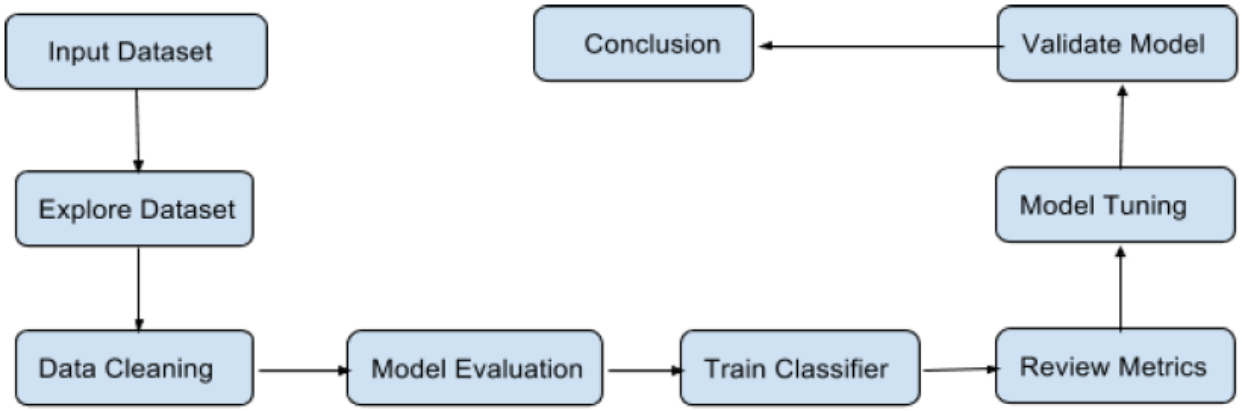
These generated columns are sometimes called dummy variables, and we will use the pandas.get_dummies () function to perform this transformation.
Several Data preprocessing steps like preprocessing feature columns, identifying feature and target columns, data cleaning and creating training and validation data splits were followed and can be referenced for details in
attached Jupyter notebook.

We also did a scaling of images to make the size of image as same for all images in order to give images as input to models.

Implementation

The project follows typical predictive analytics hierarchy as shown in Fig. 12:

Fig. 12 Project Flow

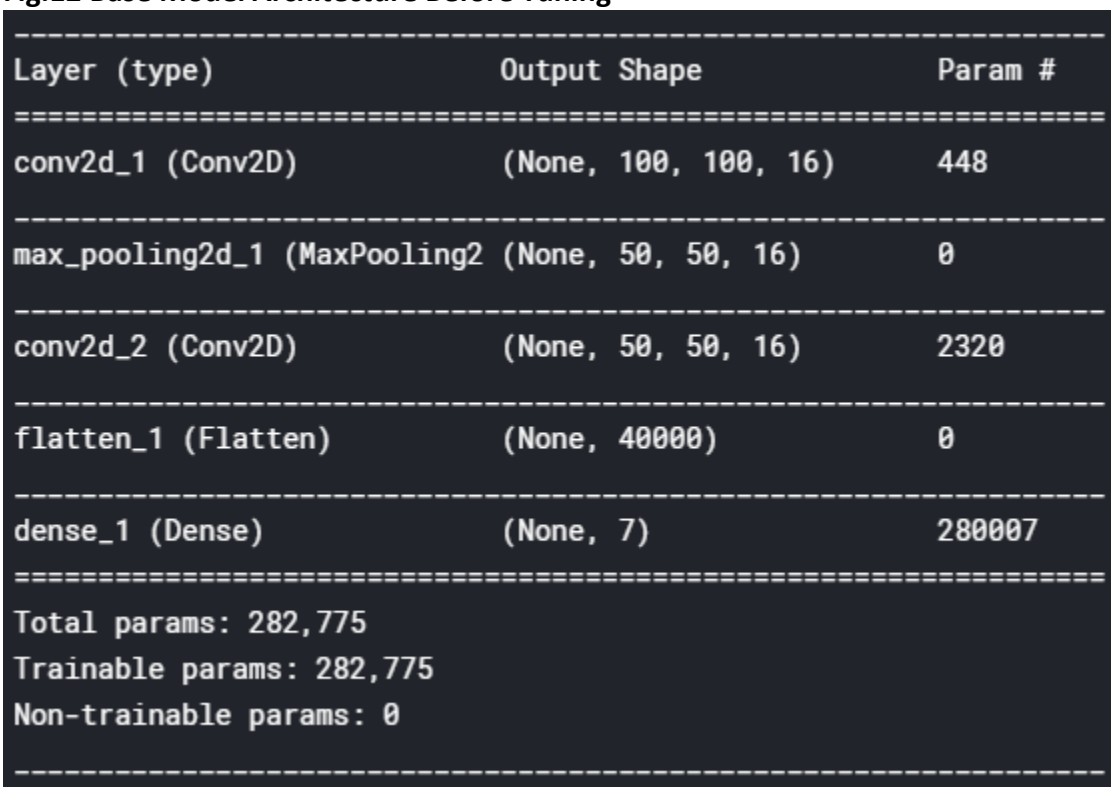


Following the direction of the arrow as shown, with the dataset we chose, we first performed exploratory analysis by seeing distribution of some of the feature that seems to be relevant for the problem we plan to solve. We also did a check of correlations using plot in this process. We then performed data cleaning steps to eliminate corrupted data's and date processing for convenience. We processed the data through a pre-processing function to scale an image of proper size and also to classify the classes and name a labels. For categorical features we converted them to dummy variables so that the dataset becomes fully numeric and it then is easy to process it through our chosen algorithm for meaningful outcome.

Before we jump into the model evaluation, we split our full dataset into training, validation and testing splits with 60:20:20 ratio and then perform another check to make sure the splits have equal percent of responses to avoid deviations in the classification iterations. We used deep learning's CNN model for training and precision, recall and f1 score metrics to perform a check on accuracy and standard error metrics to pick a model that performs best.

We first constructed a benchmark or base line model with the simple CNN architecture. For this model the model evaluation step yields around 85% accuracy. We have listed architecture for baseline model before tuning below in fig.12

Fig.12 Base Model Architecture Before Tuning



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 100, 100, 16)      448
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 50, 50, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 50, 50, 16)        2320
_________________________________________________________________
flatten_1 (Flatten)          (None, 40000)             0
_________________________________________________________________
dense_1 (Dense)              (None, 7)                 280007
=================================================================
Total params: 282,775
Trainable params: 282,775
Non-trainable params: 0
_________________________________________________________________
```

Refinement

After reviewing metrics, we started to tune the model by applying dropouts in order to avoid overfitting also gave some of the callbacks such as learning rate scheduling in order to find best learning rate and also implemented an early stopping condition in order monitor the loss error and stops the training if for a number of step given in the patience parameters the loss is not improving. We also constrained number of steps in each epoch based on the batch size in order to improve the performance of an algorithm. At last we yield a maximum accuracy of the model as around 90% which we believe is our optimized model. We have used keras library for all these implementations. We also listed the after refinement in fig.13

Fig.13 Optimized model after refinement

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 100, 100, 16)	448
max_pooling2d_2 (MaxPooling2	(None, 50, 50, 16)	0
dropout_1 (Dropout)	(None, 50, 50, 16)	0
conv2d_4 (Conv2D)	(None, 50, 50, 16)	2320
dropout_2 (Dropout)	(None, 50, 50, 16)	0
flatten_2 (Flatten)	(None, 40000)	0
dense_2 (Dense)	(None, 7)	280007

Total params: 282,775
Trainable params: 282,775
Non-trainable params: 0

The above mentioned model seems to be the best model for us to solve this problem.

IV. Results

Model Evaluation and Validation

There are several metrics available for validation but as we have discussed earlier the most suitable metrics for this problem will be precision, recall and f1 score.so using this metrics we have evaluated the classification by our model for each subspecies i.e. how many bees are classified correctly as one of the subspecies for each category of subspecies.

Thee Model Evaluation of the above given base model before refinement yields the result which is shown in the fig.13.This model gives around 85% of overall accuracy out of that a bee variety of “1 Mixed local stock 2 “classifies less accurately than others so based on this review of this model evaluation we have decided to improve the model by means of adding some techniques Which is given in the refinement steps. We also have implied model checkpoint for finding best values for the model generalizes well in the code.

Fig.14 Metrics Reviewed after base model

	precision	recall	f1-score	support
-1	0.92	0.59	0.72	74
1 Mixed local stock 2	0.56	0.47	0.51	93
Carniolan honey bee	0.98	0.97	0.97	98
Italian honey bee	0.84	0.95	0.89	621
Russian honey bee	0.95	0.95	0.95	100
VSH Italian honey bee	1.00	0.23	0.38	43
Western honey bee	1.00	1.00	1.00	6
micro avg	0.85	0.85	0.85	1035
macro avg	0.89	0.74	0.78	1035
weighted avg	0.85	0.85	0.84	1035

Loss function: 0.3333283302144728, accuracy: 0.851207729468599

After model refinement we have validated the model, this time the accuracy is improved we can see the results in the fig.15.The accuracy is around 90% and also the recall and f1 score improved for “1 Mixed local stock 2 “so we have decided that this is going to be the optimized model for us.

Fig.15 Performance after refinement

	precision	recall	f1-score	support
-1	0.92	0.82	0.87	74
1 Mixed local stock 2	0.59	0.89	0.71	93
Carniolan honey bee	0.87	1.00	0.93	98
Italian honey bee	0.98	0.89	0.93	621
Russian honey bee	0.92	0.99	0.95	100
VSH Italian honey bee	0.97	0.84	0.90	43
Western honey bee	1.00	1.00	1.00	6
micro avg	0.90	0.90	0.90	1035
macro avg	0.89	0.92	0.90	1035
weighted avg	0.92	0.90	0.91	1035

Loss function: 0.24663296658635717, accuracy: 0.9024154588796091

Justification

There may be a room for improvement on the final results, the tuned final model has made an improvement over the untuned model. There could be more ways we could improve the score by trying different CNN architecture are by using predefined models like VGG, restnet etc. But we stick to this results because this is a fair enough model based on the classification results and model evaluation so we don't want to experiment with this.

Accuracy score	Benchmark Model	Final optimized model
	0.8512	0.9024

V. Conclusion

Free-Form Visualization

We have the plot for how the accuracy and loss for training and validation set has been improved over number of epochs which is shown below.

Fig.16 Before Refining for 5 epochs

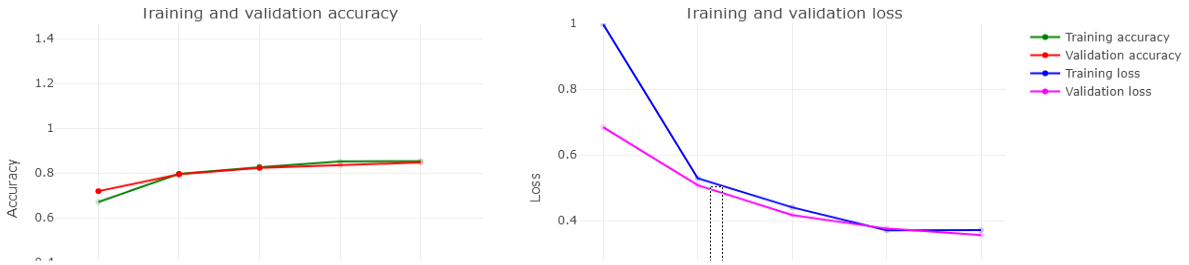
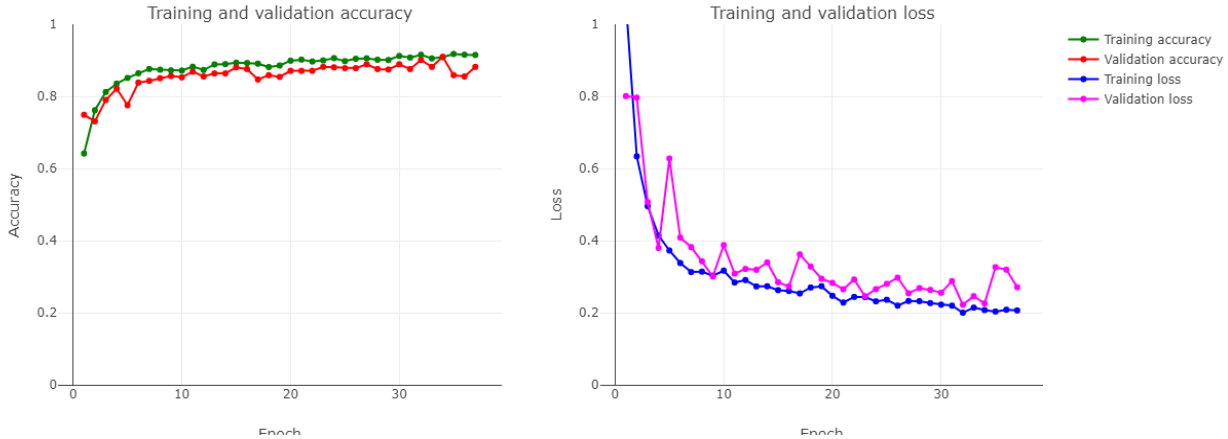


Fig.16 After Refining for 50 epochs



The difference between the number of epoch and importance about the refinement have been

clearly depicted in the above plots. This plots shows how the quality of refined model learns over number of steps.

Reflection

The most important and time consuming part of the problem was data exploration and processing as there were multiple columns with huge number of data's it is much difficult to explore each feature and try to find the imbalance between the data's. Once the data was prepared and ready, the next challenge was to pick an CNN architecture that could be best suited for the problem we choose to solve. With experience and prior knowledge and also the outcome of accuracy on training data, we observed that the simple model itself performed well for this problem but the difficult part is improving the simple model by trying different values and techniques and make them run. Selecting batch size, dropout values filters and kernel dimension was challenging.

Since there were huge number of data's and feature it is time consuming process to bring the odd one one out of the feature.so I would say both Model Refinement and exploration of data for preprocessing is much challenging.

The key lessons learned from this Kernel are the following:

- start by analyzing the data;
- follow with a simple baseline model;
- refine gradually the model, by making corrections based on the analysis of the (partial) results.

Improvements

One of the improvements we could do was to perform tuning to improve recall rate to improve overall prediction performance of the model. The other improvement could be to work with a predefined model like resnet, VGG etc. Another key aspect would be to review metrics like log-loss to understand how quick the model is able to tune.

References

[1] Gabriel Preda, RSNA Pneumonia Detection EDA, https://www.kaggle.com/gpreda/rsna-pneumonia-detection-eda

[2] Gabriel Preda, CNN with Tensorflow|Keras for Fashion-MNIST, https://www.kaggle.com/gpreda/cnn-with-tensorflow-keras-for-fashion-mnist

[3] DanB, CollinMoris, Deep Learning From Scratch, https://www.kaggle.com/dansbecker/deep-learning-from-scratch

[4] DanB, Dropout and Strides for Larger Models, https://www.kaggle.com/dansbecker/dropout-and-strides-for-larger-models

[5] BGO, CNN with Keras, https://www.kaggle.com/bugraokcu/cnn-with-keras

[6] Dmitri Pukhov, Hony Bee health detection using CNN, https://www.kaggle.com/gpreda/honey-bee-health-detection-with-cnn/notebook

[7] Why Dropounts prevent overfitting in Deep Neural Networks, https://medium.com/@vivek.yadav/why-dropouts-prevent-overfitting-in-deep-neural-networks-937e2543a701
[8] Dropout: A Simple Way to Prevent Neural Networks from Overfitting, https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf