VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

J Component report

Programme : **Integrated MTech in CSE with Specialization in Business Analytics**

Course Title : **Deep Learning**

Course Code : **CSE4037**

Slot : **G2**

TITLE

Honeybee Sub-Species and Health Classification Using Deep Learning

Team Members:

Vamsidhar S | 19MIA1074

Mansoor khan lodi | 19MIA1094

Shashank R | 19MIA1105

Faculty: Prof. Khadar Nawas K.

Sign:

Date: 29/4/22

INDEX

S.NO.	CONTENT	PAGE NO.
1	ACKNOWLEDGEMENT	3
2	ABSTRACT	4
3	INTRODUCTION	5
4	ABOUT THE DATASET	6
5	OBJECTIVE	7
6	ARCHITECTURE DIAGRAM	8
7	CODE SCREENSHOTS	11
8	RESULTS AND CONCLUSIONS	25
9	REFERENCES	26

ACKNOWLEDGMENT

Primarily, we would like to thank the almighty for all the blessings he showered over us to complete this project without any flaws.

The success and final outcome of this assignment required a lot of guidance and assistance from many people, and we are extremely fortunate to have got this all along with the completion of our project. Whatever we have done is only due to such guidance and assistance by our faculty, Prof. Khadar Nawas K, to whom we are really thankful for giving us an opportunity to do this project.

Last but not the least, we are grateful to all our fellow classmates and our friends for the suggestions and support given to us throughout the completion of our project.

ABSTRACT

Up to one third of the global food production depends on the pollination of honeybees which makes them an important role in the ecosystem. We need to have healthy honeybees in order to make best use out of it. 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 bee's workflow and hive in general. By investigating the bees that leave the hive, we can gain a more complete understanding of the hive itself. For example, an unhealthy hive infected with varroa mites will have bees with deformed wings or mites on their backs. These characteristics can be observed without opening the hive.

To enhance the process of apiculture, we will be using CNN model to detect the health of the honeybees. Along with that we will be using the same CNN model to detect the Sub-Species. After completion of the training, we will be testing our model using validation data through various plots.

INTRODUCTION

Honey is being widely used in our everyday life. It has high impact on our health and resolves many health-related issues such as healing wounds, indigestion, sore throat, and cough. It is also used as an ingredient in most of the FMCG goods such as food and beverages, cosmetics and many more.

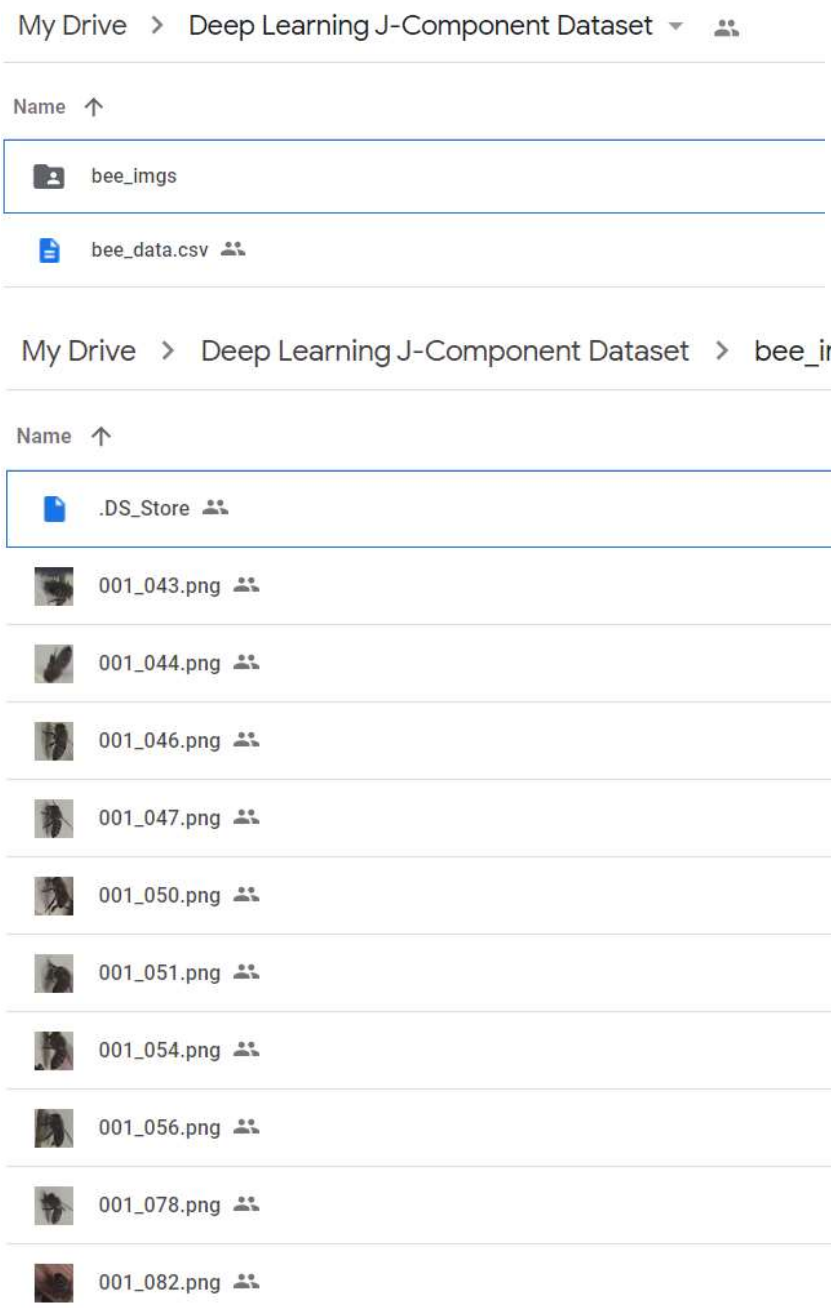
The current world is meeting with high demands of honey, due to which there is a lack of honey production. In order to improve honey production, nomadic beekeepers move their hives according to the floral blossom. The remoteness of the apiaries can make it harder for them to keep control over the hives and to intervene on time in case of illness or other stressors that affect bee health.

Bees may be small, but they play an enormous part in supporting biodiversity and maintaining natural ecosystems. They produce honey and are essential for pollinating fruit trees and crops, contributing to crop yields. Many beekeepers move their hives according to the blooming periods of flowers in the area. For them, keeping track of the remote hives can be a time consuming and costly venture.

4. ABOUT THE DATASET

The Dataset we had referred with the help of Kaggle platform and National Geography. It consists of 5100+ images and its annotations such as location, date, time, sub-species and health condition which is in the form of .CSV

https://drive.google.com/drive/folders/1q_ZfDWVJCgs8sDc5gRybNc908l7D_lCG?usp=sharing



bee_data.csv Open with

	A	B	C	D	E	F	G	H	I
1	file	date	time	location	zip code	subspecies	health	pollen_carrying	caste
2	041_066.png	8/28/18	16:07	Alvin, TX, USA	77511	-1	hive being robbed	FALSE	worker
3	041_072.png	8/28/18	16:07	Alvin, TX, USA	77511	-1	hive being robbed	FALSE	worker
4	041_073.png	8/28/18	16:07	Alvin, TX, USA	77511	-1	hive being robbed	FALSE	worker
5	041_067.png	8/28/18	16:07	Alvin, TX, USA	77511	-1	hive being robbed	FALSE	worker
6	041_059.png	8/28/18	16:07	Alvin, TX, USA	77511	-1	hive being robbed	FALSE	worker
7	041_071.png	8/28/18	16:07	Alvin, TX, USA	77511	-1	hive being robbed	FALSE	worker
8	041_065.png	8/28/18	16:07	Alvin, TX, USA	77511	-1	hive being robbed	FALSE	worker
9	041_064.png	8/28/18	16:07	Alvin, TX, USA	77511	-1	hive being robbed	FALSE	worker
10	041_070.png	8/28/18	16:07	Alvin, TX, USA	77511	-1	hive being robbed	FALSE	worker
11	041_058.png	8/28/18	16:07	Alvin, TX, USA	77511	-1	hive being robbed	FALSE	worker
12	041_074.png	8/28/18	16:07	Alvin, TX, USA	77511	-1	hive being robbed	FALSE	worker
13	041_060.png	8/28/18	16:07	Alvin, TX, USA	77511	-1	hive being robbed	FALSE	worker
14	041_048.png	8/28/18	16:07	Alvin, TX, USA	77511	-1	hive being robbed	FALSE	worker
15	041_049.png	8/28/18	16:07	Alvin, TX, USA	77511	-1	hive being robbed	FALSE	worker
16	041_061.png	8/28/18	16:07	Alvin, TX, USA	77511	-1	hive being robbed	FALSE	worker
17	041_075.png	8/28/18	16:07	Alvin, TX, USA	77511	-1	hive being robbed	FALSE	worker
18	041_063.png	8/28/18	16:07	Alvin, TX, USA	77511	-1	hive being robbed	FALSE	worker

5. OBJECTIVE/PROBLEM STATEMENT

In the present world there is a huge demand of honey, due to which there is a lack of honey production. In order to improve honey production, nomadic beekeepers move their hives according to the floral blossom. The remoteness of the apiaries can make it harder for them to keep control over the hives and to intervene on time in case of illness or other stressors that affect bee health. To avoid that complexity, we will be proposing a CNN architecture that helps in classifying subspecies and health status.

6. 1. PROPOSED ARCHITECTURE FOR BEE SUB-SPECIES CLASSIFICATION

Bees are classified based on Sub-Species. These are:

1. **Carniolan honeybee:** It's a sub-species of a western honeybee and a native to Slovenia.

2. **Italian honeybee:** It's a sub-species of a western honeybee and a native to Italy.

3. **Russian honeybee:** It's a sub-species that originates in the Primorsky region of Russia.

4. **VSH Italian honeybee:** It's a trait of honeybee that enables a colony to survive without mite controls. Its originations are from Italy.

5. **Western honeybee:** It's a European honeybee that consists of 7-12 species worldwide.

We will be building an architecture that helps in classifying these sub-species. The architecture can be seen in the below Figure (1).

Layer	filters	kernel size	Stride	Padding	Size of Feature map	Activation Function
Input					100 x 100 x3	
Conv1	12	3	-	same	100 x 100 x 6	Relu
MaxPool1	-	-	2	-	50 x 50 x 6	-
Conv2	12	3	-	same	50 x 50 x 12	Relu
MaxPool2	-	-	2	-	25 x 25 x 12	-
Conv3	12	3	-	same	25 x 25 x 12	Relu
MaxPool3	-	-	2	-	12 x 12 x 12	-
Conv4	12	3	-	same	12 x 12 x 12	Relu
MaxPool4	-	-	2	-	6 x 6 x 12	-
Conv5	12	3	-	same	6 x 6 x 12	Relu
MaxPool5	-	-	2	-	3 x 3 x 12	-
Flatten	-	-	-	-	(None ,108)	-
Dense	-	-	-	-	(None ,7)	Softmax

Figure (1)

Our proposed CNN for sub-species classification is a customized architecture consisting of five convolution layers and five max pool layers with some hyperparameters mentioned. Convolution operations are performed to obtain reduced size. Pooling is done so that we will be able to do upsampling or downsampling. We use a flatten layer to get the output in a column vector which will be useful for triggering to the next fully connected layers. There is one fully connected layer in our architecture and some hyperparameters are also mentioned for that.

6. 2. PROPOSED ARCHITECTURE FOR BEE HEALTH CLASSIFICATION

Bees are classified based on health. These are:

1. **Healthy:** These honeybees that are not suffering from any disease.

2. **Varrora mites:** These are tiny red brown external parasites of honeybees.

3. **Ant problems:** These ants colonize the honey and decreases the bee morale to fight against it.

4. **Hive being robbed:** It refers to bee killing bees.

5. **Missing queen:** Absence of queen decreases the bee morale.

We will be building an architecture that helps in classifying these bees based on health. The architecture can be seen in the below Figure (2).

Layer	filters	kernel size	Stride	Padding	Size of Feature map	Activation Function
Input					100 x 100 x3	
Conv1	12	3	-	same	100 x 100 x 6	Relu
MaxPool1	-	-	2	-	50 x 50 x 6	-
Conv2	12	3	-	same	50 x 50 x 12	Relu
MaxPool2	-	-	2	-	25 x 25 x 12	-
Conv3	12	3	-	same	25 x 25 x 12	Relu
MaxPool3	-	-	2	-	12 x 12 x 12	-
Conv4	12	3	-	same	12 x 12 x 12	Relu
MaxPool4	-	-	2	-	6 x 6 x 12	-
Conv5	12	3	-	same	6 x 6 x 12	Relu
MaxPool5	-	-	2	-	3 x 3 x 12	-
Flatten	-	-	-	-	(None ,108)	-
Dense	-	-	-	-	(None ,7)	Softmax

Figure (2)

Our proposed CNN for bee health classification is a customized architecture consisting of five convolution layers and five max pool layers with some hyperparameters mentioned. Convloution operations are performed to obtain reduced size. Pooling is done so that we will be able to do upsampling or downsampling. We use a flatten layer to get the output in a column vector which will be useful for triggering to the next fully connected layers. There is one fully connected layer in our architecture and some hyperparameters are also mentioned for that.

7. CODE SCREENSHOTS

Honey Bee Sub-Species and Health Classification using Deep Learning

Imports

In [ ]:
```
# Libraries
import pandas as pd
import numpy as np
import sys
import os
import random
from pathlib import Path

# Image processing
import imageio
import skimage
import skimage.io
import skimage.transform

# Charts
import matplotlib.pyplot as plt
import seaborn as sns


# ML
import scipy
from sklearn.model_selection import train_test_split
from sklearn import metrics


from keras import optimizers
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten, MaxPool2D, Dropout, BatchNormalization,Leak
yReLU
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ModelCheckpoint, Callback, EarlyStopping, ReduceLROnPlateau
from keras.utils import to_categorical
import tensorflow


np.random.seed(42)
tensorflow.set_random_seed(42)

# Global variables
img_folder='../input/bee_imgs/bee_imgs/'
img_width=100
img_height=100
img_channels=3
```

```
Using TensorFlow backend.
```

Reading Bee data

In [ ]:
```
bees=pd.read_csv('../input/bee_data.csv',
                 index_col=False,
                 parse_dates={'datetime':[1,2]},
                 dtype={'subspecies':'category', 'health':'category','caste':'category'})


def read_img(file):

    img = skimage.io.imread(img_folder + file)
    img = skimage.transform.resize(img, (img_width, img_height), mode='reflect')
    return img[:,:,:img_channels]


bees.dropna(inplace=True)

# Some image files don't exist. We will leave only bees with available images.
img_exists = bees['file'].apply(lambda f: os.path.exists(img_folder + f))
bees = bees[img_exists]

bees.head()
```

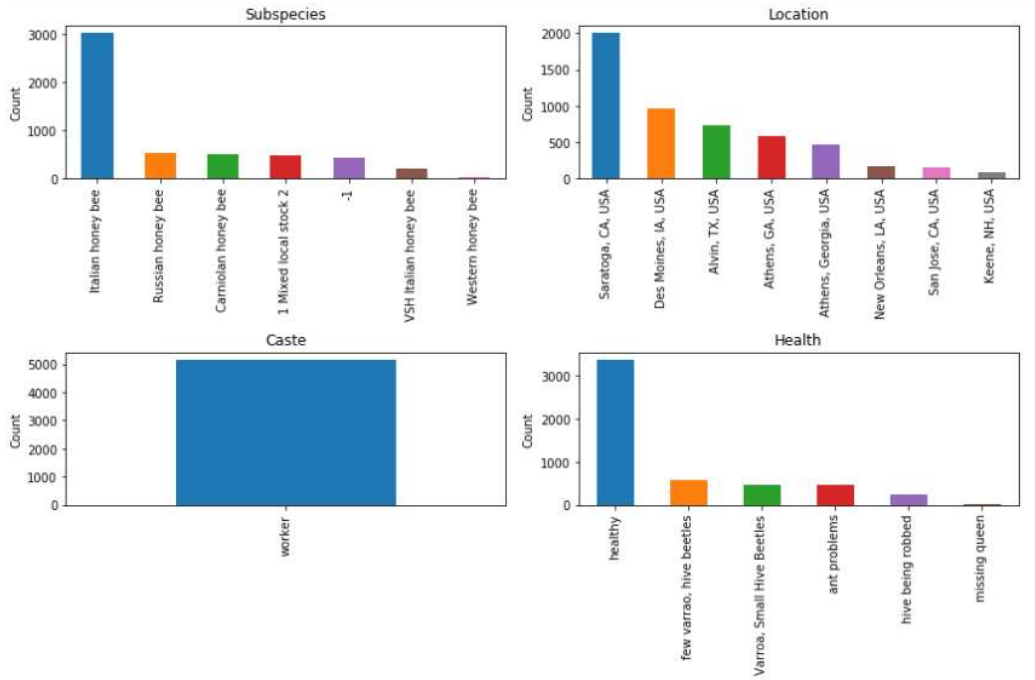
Out[ ]:

	datetime	file	location	zip code	subspecies	health	pollen_carrying	caste
0	2018-08-28 16:07:00	041_066.png	Alvin, TX, USA	77511	-1	hive being robbed	False	worker
1	2018-08-28 16:07:00	041_072.png	Alvin, TX, USA	77511	-1	hive being robbed	False	worker
2	2018-08-28 16:07:00	041_073.png	Alvin, TX, USA	77511	-1	hive being robbed	False	worker
3	2018-08-28 16:07:00	041_067.png	Alvin, TX, USA	77511	-1	hive being robbed	False	worker
4	2018-08-28 16:07:00	041_059.png	Alvin, TX, USA	77511	-1	hive being robbed	False	worker

Performing EDA

(i) Distribution of bees by categories

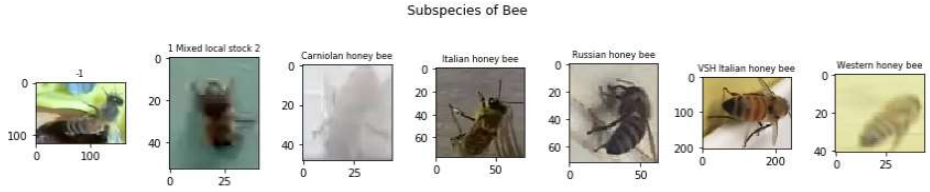
In [ ]:
```
f, ax = plt.subplots(nrows=2, ncols=2, figsize=(12,8))

bees.subspecies.value_counts().plot(kind='bar',ax=ax[0, 0])
ax[0,0].set_ylabel('Count')
ax[0,0].set_title('Subspecies')

bees.location.value_counts().plot(kind='bar', ax=ax[0, 1])
ax[0,1].set_title('Location')
ax[0,1].set_ylabel('Count')

bees.caste.value_counts().plot(kind='bar', ax=ax[1, 0])
ax[1,0].set_title('Caste')
ax[1,0].set_ylabel('Count')

bees.health.value_counts().plot(kind='bar', ax=ax[1,1])
ax[1,1].set_title('Health')
ax[1,1].set_ylabel('Count')

f.subplots_adjust(hspace=0.7)
f.tight_layout()
plt.show()
```



Bees images

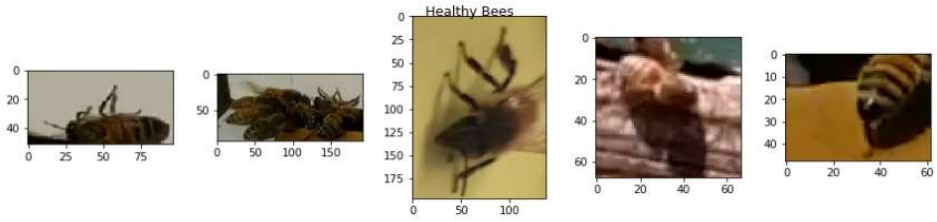
Subspecies of Bee

In [ ]:
```
# Selecting first subspecies titles
subspecies = bees['subspecies'].cat.categories
f, ax = plt.subplots(nrows=1,ncols=subspecies.size, figsize=(12,3))
i=0

# We will try to draw the first found bee of given subpecies
for s in subspecies:
    if s == 'healthy': continue
    file=img_folder + bees[bees['subspecies']==s].iloc[0]['file']
    im=imageio.imread(file)
    ax[i].imshow(im, resample=True)
    ax[i].set_title(s, fontsize=8)
    i+=1

plt.suptitle("Subspecies of Bee")
plt.tight_layout()
plt.show()
```



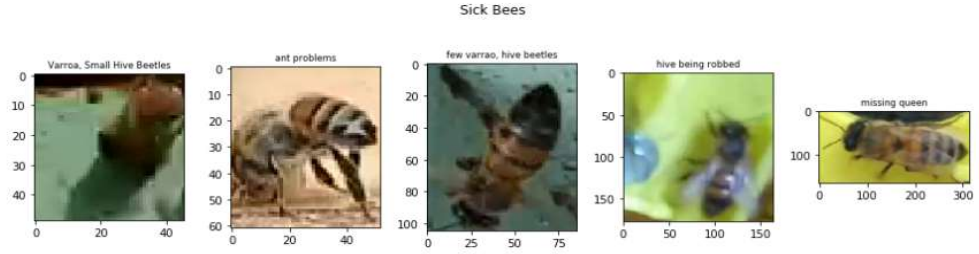
Healthy Bees

In [ ]:
```
ncols = 5
healthy = bees[bees['health'] == 'healthy'].sample(ncols)

f, ax = plt.subplots(nrows=1,ncols=ncols, figsize=(12,3))

for i in range(0,5):
    file = img_folder + healthy.iloc[i]['file']
    ax[i].imshow(imageio.imread(file))

plt.suptitle("Healthy Bees")
plt.tight_layout()
plt.show()
```



Sick Bees

In [ ]:
```
health_cats = bees['health'].cat.categories
f, ax = plt.subplots(1, health_cats.size-1, figsize=(12,4))
i=0
for c in health_cats:
    if c == 'healthy': continue
    bee = bees[bees['health'] == c].sample(1).iloc[0]
    ax[i].imshow(imageio.imread(img_folder + bee['file']))
    ax[i].set_title(bee['health'], fontsize=8)
    i += 1

plt.suptitle("Sick Bees")
plt.tight_layout()
plt.show()
```



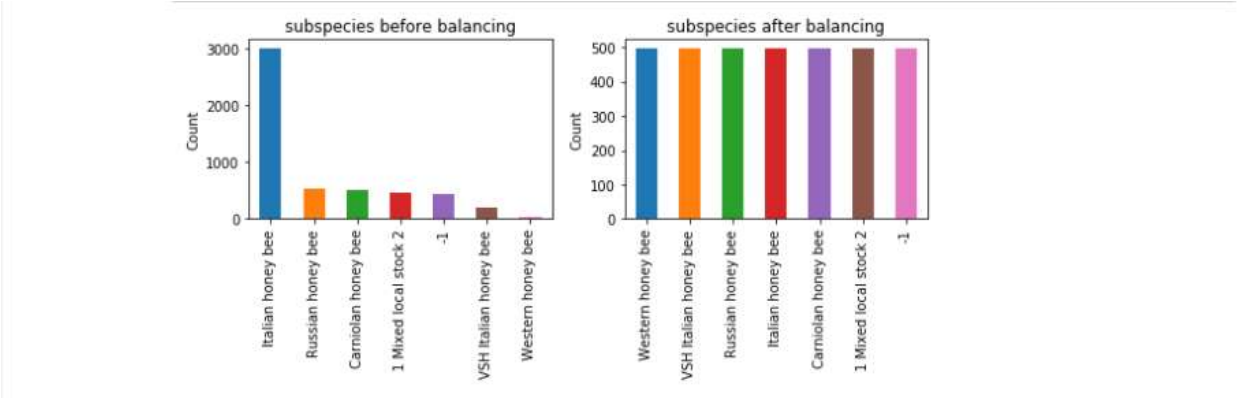
Bee subspecies classification

Preprocessing includes data balancing and augmentation. Then we'll be ready to train CNN.

(i) Data preprocessing for Bee subspecies

- Balancing samples by subspecies

In [ ]:
```
# The same split-balance will be used in 2 places: subspecies and health CNN.

def split_balance(bees, field_name):

    # Splitting to train and test before balancing
    train_bees, test_bees = train_test_split(bees, random_state=24)

    # Splitting train to train and validation datasets
    # Validation for use during learning
    train_bees, val_bees = train_test_split(train_bees, test_size=0.1, random_state=24)

    #Balance by subspecies to train_bees_bal_ss dataset
    # Number of samples in each category
    ncat_bal = int(len(train_bees)/train_bees[field_name].cat.categories.size)
    train_bees_bal = train_bees.groupby(field_name, as_index=False).apply(lambda g:  g.sampl
e(ncat_bal, replace=True)).reset_index(drop=True)
    return(train_bees_bal, val_bees, test_bees)

def plot_balanced(train_bees, train_bees_bal, field_name):

    # Plot before and after balancing
    f, axs = plt.subplots(1,2, figsize=(8,4))

    # Before
    ax = train_bees[field_name].value_counts().plot(kind='bar', ax=axs[0])
    ax.set_title('%s before balancing' % field_name)
    ax.set_ylabel('Count')

    # After
    ax = train_bees_bal[field_name].value_counts().plot(kind='bar', ax=axs[1])
    ax.set_title('%s after balancing' % field_name)
    ax.set_ylabel('Count')

    plt.tight_layout()
    plt.show()

# Split/balance and plotting the result
train_bees_bal, val_bees, test_bees = split_balance(bees, 'subspecies')
plot_balanced(bees, train_bees_bal, 'subspecies')

# Using balanced dataset as main
train_bees = train_bees_bal
```



- Preparing features

We prepare train_bees dataset to load them and use ImageDataGenerator to randomly shift/rotate/zoom.

In [ ]:
```
def prepare2train(train_bees, val_bees, test_bees, field_name):

    # Train data
    train_X = np.stack(train_bees['file'].apply(read_img))

    train_y  = pd.get_dummies(train_bees[field_name], drop_first=False)

    # Validation during training data to calc val_loss metric
    val_X = np.stack(val_bees['file'].apply(read_img))

    val_y = pd.get_dummies(val_bees[field_name], drop_first=False)

    # Test data
    test_X = np.stack(test_bees['file'].apply(read_img))

    test_y = pd.get_dummies(test_bees[field_name], drop_first=False)

    # Data augmentation - rotate, zoom and shift input images.
    generator = ImageDataGenerator(
            featurewise_center=False,
            samplewise_center=False,
            featurewise_std_normalization=False,
            samplewise_std_normalization=False,
            zca_whitening=False,
            rotation_range=180,
            zoom_range = 0.1,
            width_shift_range=0.2,
            height_shift_range=0.2,
            horizontal_flip=True,
            vertical_flip=True)
    generator.fit(train_X)
    return (generator, train_X, val_X, test_X, train_y, val_y, test_y)

# Call image preparation and one hot encoding
generator, train_X, val_X, test_X, train_y, val_y, test_y = prepare2train(train_bees, val_be
es, test_bees, 'subspecies')
```

```
/opt/conda/lib/python3.6/site-packages/skimage/transform/_warps.py:110: UserWarning: Anti-ali
asing will be enabled by default in skimage 0.15 to avoid aliasing artifacts when down-sampli
ng images.
  warn("Anti-aliasing will be enabled by default in skimage 0.15 to "
```

Train Bee Subspecies CNN

In [ ]:
```
# We'll stop training if no improvement after some epochs
earlystopper1 = EarlyStopping(monitor='loss', patience=10, verbose=1)

# Save the best model during the traning
checkpointer1 = ModelCheckpoint('best_model1.h5'
                                ,monitor='val_acc'
                                ,verbose=1
                                ,save_best_only=True
                                ,save_weights_only=True)
# Build CNN model
model1=Sequential()
model1.add(Conv2D(6, kernel_size=3, input_shape=(img_width, img_height,3), activation='relu'
, padding='same'))
model1.add(MaxPool2D(2))
model1.add(Conv2D(12, kernel_size=3, activation='relu', padding='same'))
model1.add(MaxPool2D(2))
model1.add(Conv2D(12, kernel_size=3, activation='relu', padding='same'))
model1.add(MaxPool2D(2))
model1.add(Conv2D(12, kernel_size=3, activation='relu', padding='same'))
model1.add(MaxPool2D(2))
model1.add(Conv2D(12, kernel_size=3, activation='relu', padding='same'))
model1.add(MaxPool2D(2))
model1.add(Flatten())
model1.add(Dense(train_y.columns.size, activation='softmax'))
model1.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train
training1 = model1.fit_generator(generator.flow(train_X,train_y, batch_size=60)
                        ,epochs=65
                        ,validation_data=[val_X, val_y]
                        ,steps_per_epoch=65
                        ,callbacks=[earlystopper1, checkpointer1])
# Get the best saved weights
model1.load_weights('best_model1.h5')
```

Train Bee Subspecies CNN

In [ ]:
```
# We'll stop training if no improvement after some epochs
earlystopper1 = EarlyStopping(monitor='loss', patience=10, verbose=1)

# Save the best model during the traning
checkpointer1 = ModelCheckpoint('best_model1.h5'
                                ,monitor='val_acc'
                                ,verbose=1
                                ,save_best_only=True
                                ,save_weights_only=True)
# Build CNN model
model1=Sequential()
model1.add(Conv2D(6, kernel_size=3, input_shape=(img_width, img_height,3), activation='relu'
, padding='same'))
model1.add(MaxPool2D(2))
model1.add(Conv2D(12, kernel_size=3, activation='relu', padding='same'))
model1.add(MaxPool2D(2))
model1.add(Conv2D(12, kernel_size=3, activation='relu', padding='same'))
model1.add(MaxPool2D(2))
model1.add(Conv2D(12, kernel_size=3, activation='relu', padding='same'))
model1.add(MaxPool2D(2))
model1.add(Conv2D(12, kernel_size=3, activation='relu', padding='same'))
model1.add(MaxPool2D(2))
model1.add(Flatten())
model1.add(Dense(train_y.columns.size, activation='softmax'))
model1.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train
training1 = model1.fit_generator(generator.flow(train_X,train_y, batch_size=60)
                        ,epochs=65
                        ,validation_data=[val_X, val_y]
                        ,steps_per_epoch=65
                        ,callbacks=[earlystopper1, checkpointer1])
# Get the best saved weights
model1.load_weights('best_model1.h5')
```

```
Epoch 1/65
65/65 [==============================] - 13s 200ms/step - loss: 1.8389 - acc: 0.2246 - val_lo
ss: 1.5899 - val_acc: 0.2113

Epoch 00001: val_acc improved from -inf to 0.21134, saving model to best_model1.h5
Epoch 2/65
65/65 [==============================] - 8s 130ms/step - loss: 1.3331 - acc: 0.4177 - val_los
s: 1.3108 - val_acc: 0.3170

Epoch 00002: val_acc improved from 0.21134 to 0.31701, saving model to best_model1.h5
Epoch 3/65
65/65 [==============================] - 8s 127ms/step - loss: 0.9593 - acc: 0.6161 - val_los
s: 1.1133 - val_acc: 0.4923

Epoch 00003: val_acc improved from 0.31701 to 0.49227, saving model to best_model1.h5
Epoch 4/65
65/65 [==============================] - 8s 126ms/step - loss: 0.7611 - acc: 0.7023 - val_los
s: 1.0656 - val_acc: 0.5412

Epoch 00004: val_acc improved from 0.49227 to 0.54124, saving model to best_model1.h5
Epoch 5/65
65/65 [==============================] - 8s 127ms/step - loss: 0.6544 - acc: 0.7523 - val_los
s: 1.1208 - val_acc: 0.5670

Epoch 00005: val_acc improved from 0.54124 to 0.56701, saving model to best_model1.h5
Epoch 6/65
65/65 [==============================] - 8s 124ms/step - loss: 0.6396 - acc: 0.7518 - val_los
s: 0.8074 - val_acc: 0.6804

Epoch 00006: val_acc improved from 0.56701 to 0.68041, saving model to best_model1.h5
Epoch 7/65
65/65 [==============================] - 9s 135ms/step - loss: 0.5741 - acc: 0.7838 - val_los
s: 0.9344 - val_acc: 0.6598

Epoch 00007: val_acc did not improve from 0.68041
Epoch 8/65
65/65 [==============================] - 8s 128ms/step - loss: 0.5321 - acc: 0.8049 - val_los
s: 0.7801 - val_acc: 0.6804

Epoch 00008: val_acc did not improve from 0.68041
Epoch 9/65
65/65 [==============================] - 8s 125ms/step - loss: 0.4962 - acc: 0.8256 - val_los
s: 0.7610 - val_acc: 0.6778

Epoch 00009: val_acc did not improve from 0.68041
Epoch 10/65
```

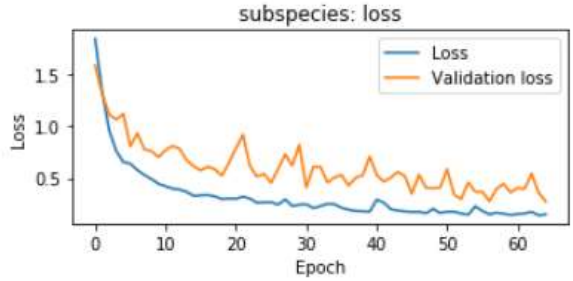
```
Epoch 00063: val_acc did not improve from 0.86340
Epoch 64/65
65/65 [==============================] - 8s 123ms/step - loss: 0.1417 - acc: 0.9505 - val_los
s: 0.3607 - val_acc: 0.8582

Epoch 00064: val_acc did not improve from 0.86340
Epoch 65/65
65/65 [==============================] - 9s 134ms/step - loss: 0.1497 - acc: 0.9464 - val_los
s: 0.2764 - val_acc: 0.8557

Epoch 00065: val_acc did not improve from 0.86340
```

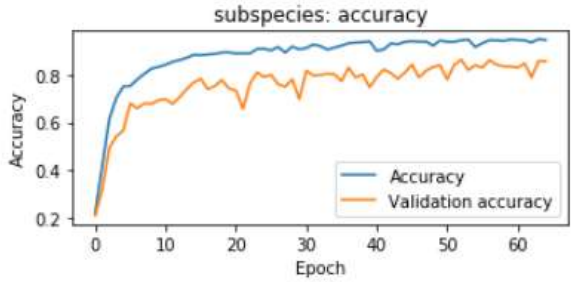
In [ ]:
```
model1.summary()
```

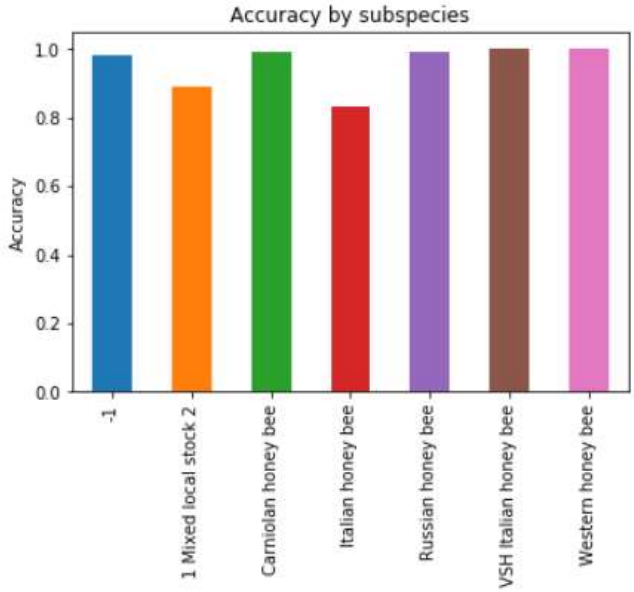
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 100, 100, 6)       168
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 50, 50, 6)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 50, 50, 12)        660
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 25, 25, 12)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 25, 25, 12)        1308
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 12, 12, 12)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 12, 12, 12)        1308
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 6, 6, 12)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 6, 6, 12)          1308
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 3, 3, 12)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 108)               0
_________________________________________________________________
dense_1 (Dense)              (None, 7)                 763
=================================================================
Total params: 5,515
Trainable params: 5,515
Non-trainable params: 0
_________________________________________________________________
```

4.3 Evaluate bee subspecies detection model

In [ ]:
```
def eval_model(training, model, test_X, test_y, field_name):

    # Trained model analysis and evaluation
    f, ax = plt.subplots(2,1, figsize=(5,5))
    ax[0].plot(training.history['loss'], label="Loss")
    ax[0].plot(training.history['val_loss'], label="Validation loss")
    ax[0].set_title('%s: loss' % field_name)
    ax[0].set_xlabel('Epoch')
    ax[0].set_ylabel('Loss')
    ax[0].legend()

    # Accuracy
    ax[1].plot(training1.history['acc'], label="Accuracy")
    ax[1].plot(training1.history['val_acc'], label="Validation accuracy")
    ax[1].set_title('%s: accuracy' % field_name)
    ax[1].set_xlabel('Epoch')
    ax[1].set_ylabel('Accuracy')
    ax[1].legend()
    plt.tight_layout()
    plt.show()

    # Accuracy by subspecies
    test_pred = model.predict(test_X)

    acc_by_subspecies = np.logical_and((test_pred > 0.5), test_y).sum()/test_y.sum()
    acc_by_subspecies.plot(kind='bar', title='Accuracy by %s' % field_name)
    plt.ylabel('Accuracy')
    plt.show()

    # Printing metrics
    print("Classification report")
    test_pred = np.argmax(test_pred, axis=1)
    test_truth = np.argmax(test_y.values, axis=1)
    print(metrics.classification_report(test_truth, test_pred, target_names=test_y.columns))

    # Loss function and accuracy
    test_res = model.evaluate(test_X, test_y.values, verbose=0)
    print('Loss function: %s, accuracy:' % test_res[0], test_res[1])

# Call evaluation function
eval_model(training1, model1, test_X, test_y, 'subspecies')
```







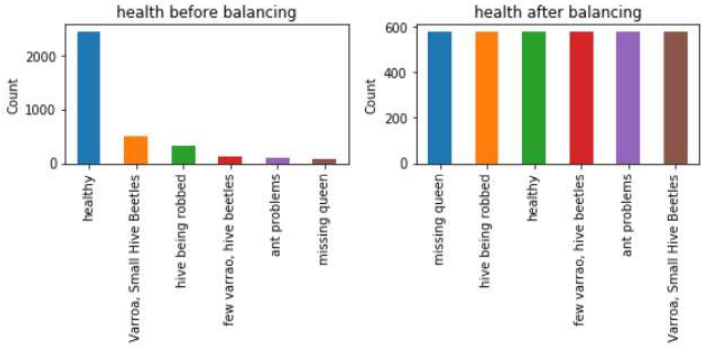

```
Classification report
                      precision    recall  f1-score   support

                  -1       0.83      0.98      0.90       108
1 Mixed local stock 2       0.49      0.89      0.63       102
  Carniolan honey bee       0.97      0.99      0.98       147
    Italian honey bee       0.98      0.83      0.90       754
    Russian honey bee       0.98      0.99      0.99       124
VSH Italian honey bee       0.90      1.00      0.95        54
    Western honey bee       1.00      1.00      1.00         4

            micro avg       0.89      0.89      0.89      1293
            macro avg       0.88      0.96      0.91      1293
         weighted avg       0.93      0.89      0.90      1293

Loss function: 0.24511571190473375, accuracy: 0.8917246713992338
```

2. CNN model for Bee health classification

Data preprocessing for Bee health classification

- Balance Bees by health

In [ ]:
```
# Split/balance and plot the result
train_bees_bal, val_bees, test_bees = split_balance(bees, 'health')
plot_balanced(train_bees, train_bees_bal, 'health')

# we will use balanced dataset as main
train_bees_bal = train_bees
```



Features augmentation and labels one hot encoding for Health CNN

In [ ]:
```
# Call image preparation and one hot encoding from Bee subspecies section
generator, train_X, val_X, test_X, train_y, val_y, test_y = prepare2train(train_bees, val_be
es, test_bees, 'health')
```

```
/opt/conda/lib/python3.6/site-packages/skimage/transform/_warps.py:110: UserWarning: Anti-ali
asing will be enabled by default in skimage 0.15 to avoid aliasing artifacts when down-sampli
ng images.
  warn("Anti-aliasing will be enabled by default in skimage 0.15 to "
```

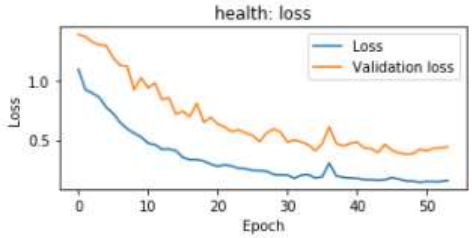
5.2 Train Bee health CNN

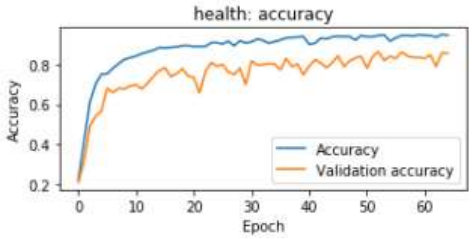
In [ ]:
```
# We'll stop training if no improvement after some epochs
earlystopper2 = EarlyStopping(monitor='val_acc', patience=10, verbose=1)

# Save the best model during the traning
checkpointer2 = ModelCheckpoint('best_model2.h5'
                                ,monitor='val_acc'
                                ,verbose=1
                                ,save_best_only=True
                                ,save_weights_only=True)
# Build CNN model
model2=Sequential()
model2.add(Conv2D(6, kernel_size=3, input_shape=(img_width, img_height,3), activation='relu'
, padding='same'))
model2.add(MaxPool2D(2))
model2.add(Conv2D(12, kernel_size=3, activation='relu', padding='same'))
model2.add(MaxPool2D(2))
model2.add(Conv2D(4, kernel_size=3, activation='relu', padding='same'))
model2.add(MaxPool2D(2))
model2.add(Conv2D(6, kernel_size=3, activation='relu', padding='same'))
model2.add(MaxPool2D(2))
model2.add(Conv2D(12, kernel_size=3, activation='relu', padding='same'))
model2.add(MaxPool2D(2))
model2.add(Flatten())
model2.add(Dense(train_y.columns.size, activation='softmax'))
model2.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train
training2 = model2.fit_generator(generator.flow(train_X,train_y, batch_size=60)
                        ,epochs=65
                        ,validation_data=[val_X, val_y]
                        ,steps_per_epoch=65
                        ,callbacks=[earlystopper2, checkpointer2])

# Get the best saved weights
model2.load_weights('best_model2.h5')
```

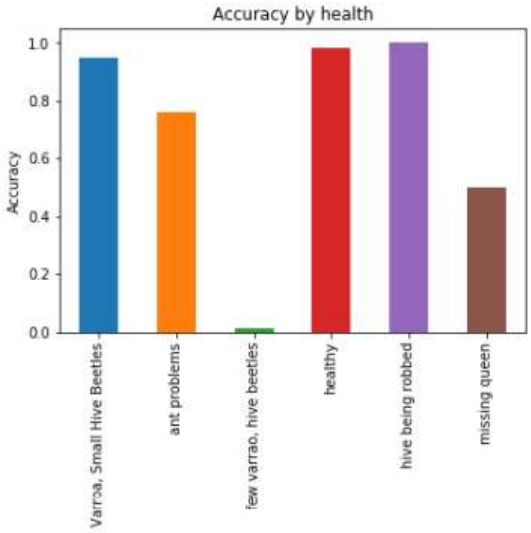
```
Epoch 1/65
65/65 [==============================] - 8s 129ms/step - loss: 1.0909 - acc: 0.6897 - val_los
s: 1.3882 - val_acc: 0.6443

Epoch 00001: val_acc improved from -inf to 0.64433, saving model to best_model2.h5
Epoch 2/65
65/65 [==============================] - 8s 128ms/step - loss: 0.9235 - acc: 0.7054 - val_los
s: 1.3701 - val_acc: 0.6443

Epoch 00002: val_acc did not improve from 0.64433
Epoch 3/65
65/65 [==============================] - 8s 126ms/step - loss: 0.8912 - acc: 0.7051 - val_los
s: 1.3242 - val_acc: 0.6418

Epoch 00003: val_acc did not improve from 0.64433
Epoch 4/65
65/65 [==============================] - 9s 135ms/step - loss: 0.8592 - acc: 0.6972 - val_los
s: 1.2994 - val_acc: 0.5825

Epoch 00004: val_acc did not improve from 0.64433
Epoch 5/65
65/65 [==============================] - 8s 121ms/step - loss: 0.7694 - acc: 0.7333 - val_los
s: 1.2929 - val_acc: 0.5593

Epoch 00005: val_acc did not improve from 0.64433
Epoch 6/65
65/65 [==============================] - 8s 128ms/step - loss: 0.7196 - acc: 0.7741 - val_los
s: 1.1892 - val_acc: 0.6211

Epoch 00006: val_acc did not improve from 0.64433
Epoch 7/65
65/65 [==============================] - 8s 125ms/step - loss: 0.6518 - acc: 0.7980 - val_los
s: 1.1270 - val_acc: 0.6340

Epoch 00007: val_acc did not improve from 0.64433
Epoch 8/65
65/65 [==============================] - 9s 135ms/step - loss: 0.5896 - acc: 0.8149 - val_los
s: 1.1191 - val_acc: 0.6392

Epoch 00008: val_acc did not improve from 0.64433
Epoch 9/65
65/65 [==============================] - 8s 121ms/step - loss: 0.5623 - acc: 0.8229 - val_los
s: 0.9237 - val_acc: 0.6881

Epoch 00009: val_acc improved from 0.64433 to 0.68814, saving model to best_model2.h5
Epoch 10/65
65/65 [==============================] - 8s 125ms/step - loss: 0.5220 - acc: 0.8315 - val_los
s: 1.0235 - val_acc: 0.6521

Epoch 00010: val_acc did not improve from 0.68814
Epoch 11/65
65/65 [==============================] - 8s 127ms/step - loss: 0.4711 - acc: 0.8451 - val_los
s: 0.9378 - val_acc: 0.6701
Epoch 00047: val_acc did not improve from 0.84021
Epoch 48/65
65/65 [==============================] - 8s 126ms/step - loss: 0.1551 - acc: 0.9469 - val_los
s: 0.3808 - val_acc: 0.8351

Epoch 00048: val_acc did not improve from 0.84021
Epoch 49/65
65/65 [==============================] - 8s 129ms/step - loss: 0.1537 - acc: 0.9441 - val_los
s: 0.3853 - val_acc: 0.8325

Epoch 00049: val_acc did not improve from 0.84021
Epoch 50/65
65/65 [==============================] - 8s 118ms/step - loss: 0.1431 - acc: 0.9495 - val_los
s: 0.4218 - val_acc: 0.8299

Epoch 00050: val_acc did not improve from 0.84021
Epoch 51/65
65/65 [==============================] - 9s 132ms/step - loss: 0.1512 - acc: 0.9451 - val_los
s: 0.4100 - val_acc: 0.8299

Epoch 00051: val_acc did not improve from 0.84021
Epoch 52/65
65/65 [==============================] - 8s 124ms/step - loss: 0.1494 - acc: 0.9508 - val_los
s: 0.4304 - val_acc: 0.8325

Epoch 00052: val_acc did not improve from 0.84021
Epoch 53/65
65/65 [==============================] - 8s 126ms/step - loss: 0.1540 - acc: 0.9444 - val_los
s: 0.4353 - val_acc: 0.8273

Epoch 00053: val_acc did not improve from 0.84021
Epoch 54/65
65/65 [==============================] - 8s 120ms/step - loss: 0.1585 - acc: 0.9454 - val_los
s: 0.4434 - val_acc: 0.8273

Epoch 00054: val_acc did not improve from 0.84021
Epoch 00054: early stopping
```

5.3 Evaluate Bee health classification model

In [ ]:
```
# Call evaluation with charts
eval_model(training2, model2, test_X, test_y, 'health')
```








```
Classification report
                           precision    recall  f1-score   support

Varroa, Small Hive Beetles       0.48      0.96      0.64       102
             ant problems       0.94      0.76      0.84       117
  few varrao, hive beetles       0.44      0.03      0.05       139
                   healthy       0.94      0.99      0.96       866
         hive being robbed       0.90      1.00      0.95        61
             missing queen       0.67      0.50      0.57         8

                 micro avg       0.86      0.86      0.86      1293
                 macro avg       0.73      0.71      0.67      1293
              weighted avg       0.85      0.86      0.83      1293

Loss function: 0.37857956102165435, accuracy: 0.8600154679501969
```

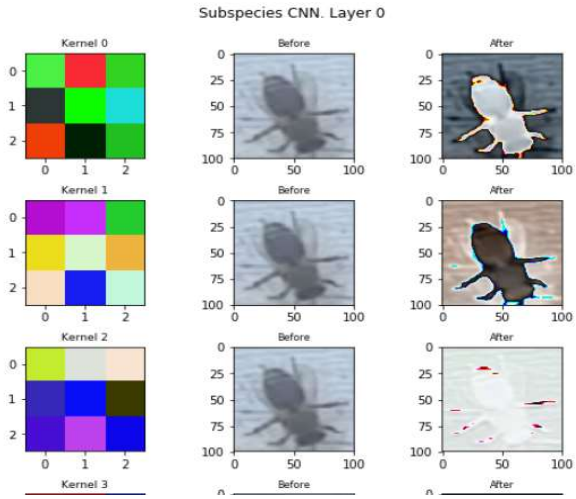
6. Visualization of Conv2D layers

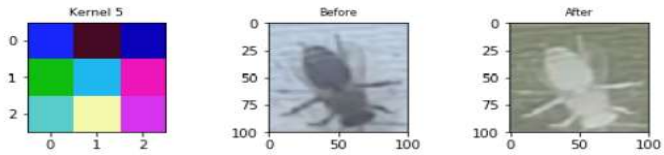
In [ ]:
```
# Now we will be looking how our models process images. Our models contains Conv2D layers wi
th kernels inside.
# We are going to convolve a sample image through kernels and see how does it look before an
d after each kernel.

# Function for Conv2D layers visualization:
```

In [ ]:
```
# Common function for visualization of kernels
def visualize_layer_kernels(img, conv_layer, title):

    # Extracting kernels from given layer
    weights1 = conv_layer.get_weights()
    kernels = weights1[0]
    kernels_num = kernels.shape[3]

    # Each row contains 3 images: kernel, input image, output image
    f, ax = plt.subplots(kernels_num, 3, figsize=(7, kernels_num*2))

    for i in range(0, kernels_num):

        kernel=kernels[:,:,:3,i]
        ax[i][0].imshow((kernel * 255).astype(np.uint8), vmin=0, vmax=255)
        ax[i][0].set_title("Kernel %d" % i, fontsize = 9)

        # Getting and draw sample image from test data
        ax[i][1].imshow((img * 255).astype(np.uint8), vmin=0, vmax=255)
        ax[i][1].set_title("Before", fontsize=8)

        # Filtered image - apply convolution
        img_filt = scipy.ndimage.filters.convolve(img, kernel)
        ax[i][2].imshow((img_filt * 255).astype(np.uint8), vmin=0, vmax=255)
        ax[i][2].set_title("After", fontsize=8)

    plt.suptitle(title)
    plt.tight_layout()
    plt.subplots_adjust(top=0.93)
    plt.show()
```

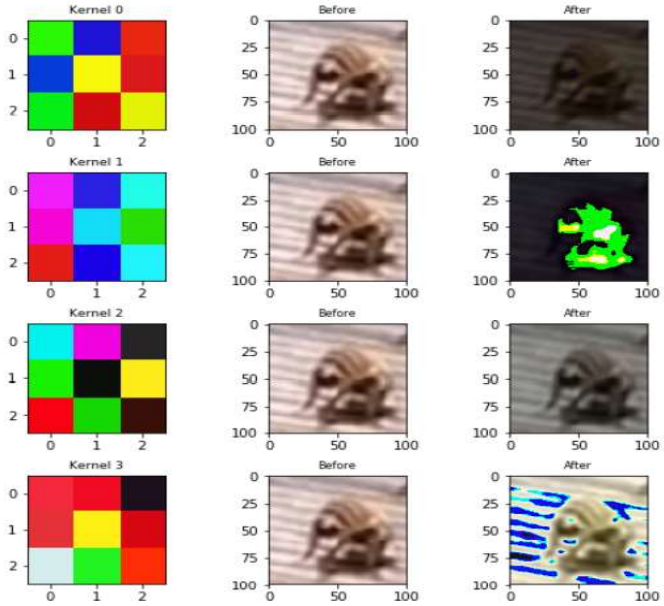
Visualizing convolutions in Bee subspecies CNN

In [ ]:
```
# Taking a sample image to visualize convolution
idx = random.randint(0,len(test_X)-1)
img = test_X[idx,:,:,:]

# Taking 1st convolutional layer and we will look at it's filters
conv1 = model1.layers[0]
img = visualize_layer_kernels(img, conv1, "Subspecies CNN. Layer 0")

# Taking sample image to visualize convolution
idx = random.randint(0,len(test_y)-1)
img = test_X[idx,:,:,:]

# Taking another convolutional layer and lwe will ook at it's filters
conv2 = model1.layers[2]
res = visualize_layer_kernels(img, conv2, "Subspecies CNN. Layer 2")
```



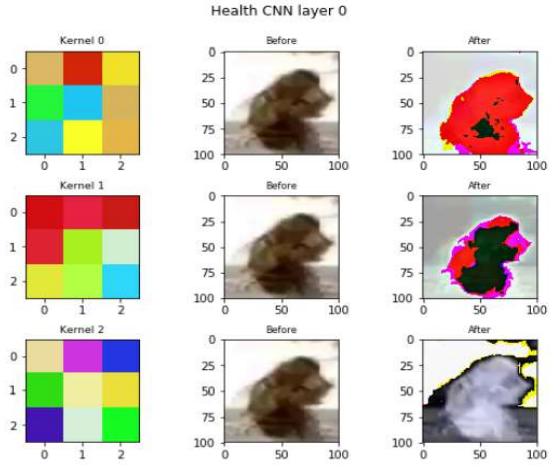


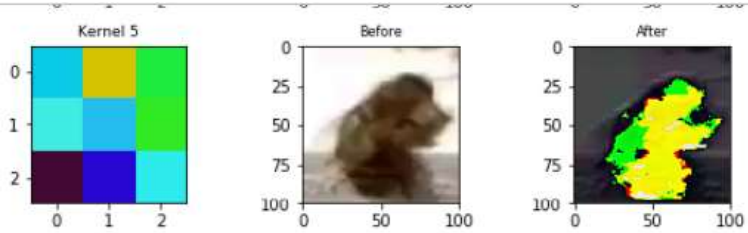
Subspecies CNN. Layer 2



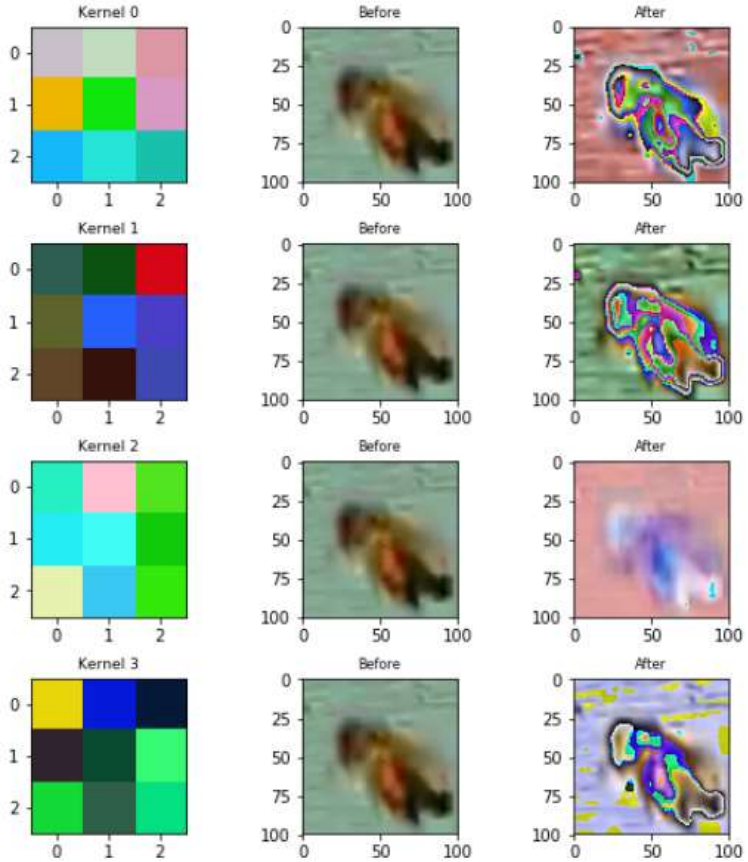
Visualizing convolutions in Bee health CNN

In [ ]:
```
# Take sample image to visualize convolution
idx = random.randint(0,len(test_X)-1)
img = test_X[idx,:,:,:]

# Taking 1st convolutional layer and look at it's filters
conv1 = model2.layers[0]
visualize_layer_kernels(img, conv1, "Health CNN layer 0")

# Taking sample image to visualize convolutoin
idx = random.randint(0,len(test_X)-1)
img = test_X[idx,:,:,:]

# Taking another convolutional layer and look at it's filters
conv2 = model2.layers[2]
visualize_layer_kernels(img, conv2, "Health CNN layer 2")
```





Health CNN layer 2



8. RESULTS AND CONCLUSION

- ***Bee Sub-Species Classification Report –***

Classification report

	precision	recall	f1-score	support
-1	0.83	0.98	0.90	108
1 Mixed local stock 2	0.49	0.89	0.63	102
Carniolan honey bee	0.97	0.99	0.98	147
Italian honey bee	0.98	0.83	0.90	754
Russian honey bee	0.98	0.99	0.99	124
VSH Italian honey bee	0.90	1.00	0.95	54
Western honey bee	1.00	1.00	1.00	4
micro avg	0.89	0.89	0.89	1293
macro avg	0.88	0.96	0.91	1293
weighted avg	0.93	0.89	0.90	1293

Loss function: 0.24511571190473375, accuracy: 0.8917246713992338

The above report talks about various classification metrics such as **precision, recall, f1-score,** and **support**. And after considering all these metrics into consideration, we can say that our model performs well, with an accuracy score of 0.891 for Bee Sub-Species Classification.

- ***Bee Health Classification Report -***

Classification report

	precision	recall	f1-score	support
Varroa, Small Hive Beetles	0.51	0.95	0.66	102
ant problems	0.86	0.79	0.83	117
few varrao, hive beetles	0.82	0.19	0.31	139
healthy	0.95	0.97	0.96	866
hive being robbed	0.91	0.98	0.94	61
missing queen	1.00	0.88	0.93	8
micro avg	0.87	0.87	0.87	1293
macro avg	0.84	0.80	0.77	1293
weighted avg	0.89	0.87	0.85	1293

Loss function: 0.34877999666257137, accuracy: 0.8708430008194932

The above report talks about various classification metrics such as **precision, recall, f1-score,** and **support**. And after considering all these metrics into consideration, we can say that our model performs well, with an accuracy score of 0.870 for Bee Health Classification.

- CNN Model for Bee Sub-Species Classification: 0.891 (89.1%)
- CNN Model for Bee Health Classification: 0.870 (87.0%)

9. REFERENCES

https://www.analyticsvidhya.com/blog/2018/12/guide-convolutional-neural-network-cnn/

https://medium.com/analytics-vidhya/cnn-32fbc1706a89

https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226

https://www.osbeehives.com/blogs/beekeeping-blog/types-of-honey-bee-and-their-traits

https://www.nationalgeographic.com/animals/invertebrates/facts/honeybee