Using Deep Learning for Detecting Soybean Diseases

Prithviraj Lakkakula

1/5/2021

## Soybean Disease Image Data

Here I use 10 images for each of the four diseases of soybean for detecting the disease using deep learning. The diseases include bacterial blight, bacterial pustule, downy mildew, and sudden death syndrome. In this document, I demonstrate the use of deep learning methods to help detect the diseases in soybeans.

Before proceeding, first we need to download and call the libraries of the following packages. Please follow the steps below.

### Step 1. Loading required R packages

library(EBImage)

## Warning: package 'EBImage' was built under R version 4.0.3

library(keras)

##   
## Attaching package: 'keras'

## The following object is masked from 'package:EBImage':  
##   
## normalize

## Read Images

#setwd('/Volumes/RAJ/DLImages/idata⁩')  
images <- c('BB1.JPG', 'BB2.JPG', 'BB3.JPG', 'BB4.JPG', 'BB5.JPG', 'BB6.JPG', 'BB7.JPG', 'BB8.JPG', 'BB9.JPG', 'BB10.JPG',  
 'BP1.JPG', 'BP2.JPG', 'BP3.JPG', 'BP4.JPG', 'BP5.JPG', 'BP6.JPG', 'BP7.JPG', 'BP8.JPG', 'BP9.JPG', 'BP10.JPG',  
 'DM1.JPG', 'DM2.JPG', 'DM3.JPG', 'DM4.JPG', 'DM5.JPG', 'DM6.JPG', 'DM7.JPG', 'DM8.JPG', 'DM9.JPG', 'DM10.JPG',  
 'SD1.JPG', 'SD2.JPG', 'SD3.JPG', 'SD4.JPG', 'SD5.JPG', 'SD6.JPG', 'SD7.JPG', 'SD8.JPG', 'SD9.JPG', 'SD10.JPG')  
  
myimages <- list()  
for (i in 1:40) {myimages[[i]] <- readImage(images[i])}

## Explore

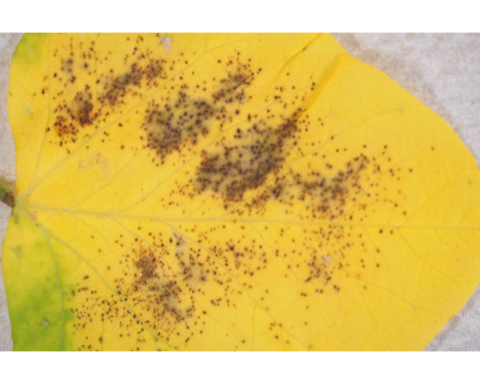
print(myimages[[1]])

## Image   
## colorMode : Color   
## storage.mode : double   
## dim : 6016 4016 3   
## frames.total : 3   
## frames.render: 1   
##   
## imageData(object)[1:5,1:6,1]  
## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] 0.4352941 0.4313725 0.4235294 0.4117647 0.4000000 0.3960784  
## [2,] 0.4392157 0.4352941 0.4274510 0.4156863 0.4039216 0.3960784  
## [3,] 0.4392157 0.4352941 0.4274510 0.4196078 0.4078431 0.4000000  
## [4,] 0.4235294 0.4235294 0.4235294 0.4196078 0.4117647 0.4078431  
## [5,] 0.4196078 0.4156863 0.4196078 0.4235294 0.4156863 0.4078431

display(myimages[[1]])



display(myimages[[11]])



display(myimages[[21]])



display(myimages[[31]])



summary(myimages[[1]])

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0000 0.3608 0.4588 0.4724 0.5765 1.0000

summary(myimages[[11]])

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0000 0.3412 0.7804 0.6575 0.8471 1.0000

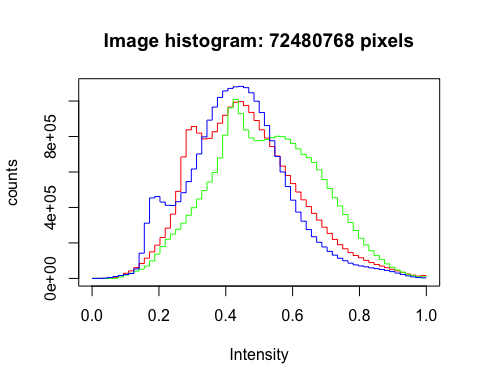
summary(myimages[[21]])

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0000 0.4431 0.5176 0.5084 0.5882 0.9961

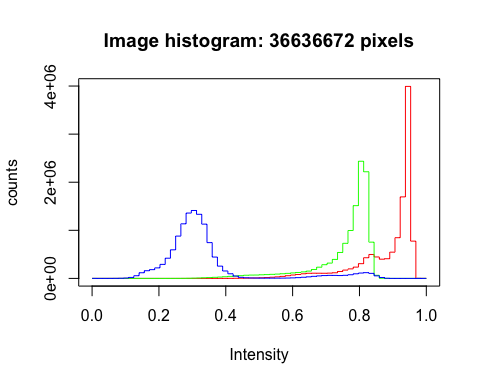
summary(myimages[[31]])

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0000 0.1059 0.2392 0.3264 0.4941 1.0000

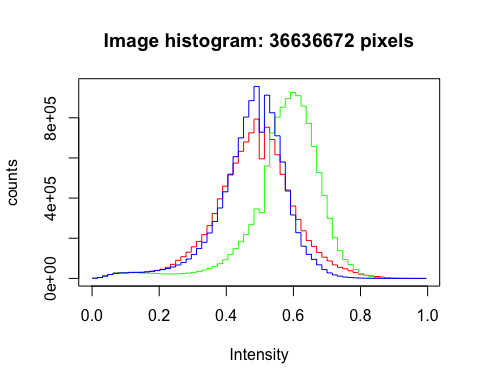
hist(myimages[[1]])



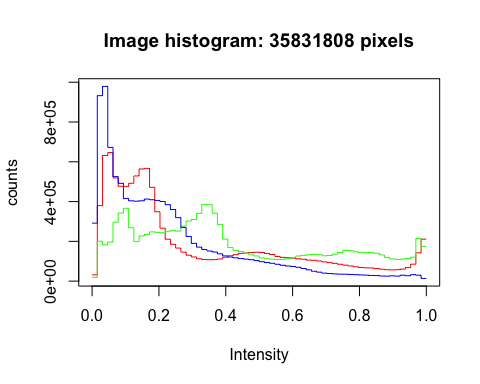
hist(myimages[[11]])



hist(myimages[[21]])



hist(myimages[[31]])



str(myimages[[1]])

## Formal class 'Image' [package "EBImage"] with 2 slots  
## ..@ .Data : num [1:6016, 1:4016, 1:3] 0.435 0.439 0.439 0.424 0.42 ...  
## ..@ colormode: int 2

str(myimages[[11]])

## Formal class 'Image' [package "EBImage"] with 2 slots  
## ..@ .Data : num [1:4288, 1:2848, 1:3] 0.792 0.788 0.788 0.788 0.788 ...  
## ..@ colormode: int 2

str(myimages[[21]])

## Formal class 'Image' [package "EBImage"] with 2 slots  
## ..@ .Data : num [1:4288, 1:2848, 1:3] 0.565 0.573 0.58 0.573 0.573 ...  
## ..@ colormode: int 2

str(myimages[[31]])

## Formal class 'Image' [package "EBImage"] with 2 slots  
## ..@ .Data : num [1:4608, 1:2592, 1:3] 0.51 0.522 0.522 0.514 0.51 ...  
## ..@ colormode: int 2

## Resizing Images

for (i in 1:40) {myimages[[i]] <- resize(myimages[[i]], 28, 28)}

## Reshaping Images

for (i in 1:40) {myimages[[i]] <- array\_reshape(myimages[[i]], c(28, 28, 3))}

## Row bind

x.train <- NULL  
for (i in 1:8) {x.train <- rbind(x.train, myimages[[i]])}  
for (i in 11:18) {x.train <- rbind(x.train, myimages[[i]])}  
for (i in 21:28) {x.train <- rbind(x.train, myimages[[i]])}  
for (i in 31:38) {x.train <- rbind(x.train, myimages[[i]])}  
  
str(x.train)

## num [1:32, 1:2352] 0.345 0.384 0.244 0.308 0.393 ...

x.test <- NULL  
for (i in 9:10) {x.test <- rbind(x.test, myimages[[i]])}  
for (i in 19:20) {x.test <- rbind(x.test, myimages[[i]])}  
for (i in 29:30) {x.test <- rbind(x.test, myimages[[i]])}  
for (i in 39:40) {x.test <- rbind(x.test, myimages[[i]])}  
#or you can do this as below as well  
#x.test <- rbind(myimages[[9]], myimages[[10]], myimages[[19]], myimages[[20]], myimages[[29]], myimages[[30]], myimages[[39]], myimages[[40]])  
str(x.test)

## num [1:8, 1:2352] 0.26 0.568 0.788 0.789 0.555 ...

y.train <- c(0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3)  
y.test <- c(0,0,1,1,2,2,3,3)

## One Hot Encoding

train.labels <- to\_categorical(y.train)  
test.labels <- to\_categorical(y.test)  
  
train.labels

## [,1] [,2] [,3] [,4]  
## [1,] 1 0 0 0  
## [2,] 1 0 0 0  
## [3,] 1 0 0 0  
## [4,] 1 0 0 0  
## [5,] 1 0 0 0  
## [6,] 1 0 0 0  
## [7,] 1 0 0 0  
## [8,] 1 0 0 0  
## [9,] 0 1 0 0  
## [10,] 0 1 0 0  
## [11,] 0 1 0 0  
## [12,] 0 1 0 0  
## [13,] 0 1 0 0  
## [14,] 0 1 0 0  
## [15,] 0 1 0 0  
## [16,] 0 1 0 0  
## [17,] 0 0 1 0  
## [18,] 0 0 1 0  
## [19,] 0 0 1 0  
## [20,] 0 0 1 0  
## [21,] 0 0 1 0  
## [22,] 0 0 1 0  
## [23,] 0 0 1 0  
## [24,] 0 0 1 0  
## [25,] 0 0 0 1  
## [26,] 0 0 0 1  
## [27,] 0 0 0 1  
## [28,] 0 0 0 1  
## [29,] 0 0 0 1  
## [30,] 0 0 0 1  
## [31,] 0 0 0 1  
## [32,] 0 0 0 1

test.labels

## [,1] [,2] [,3] [,4]  
## [1,] 1 0 0 0  
## [2,] 1 0 0 0  
## [3,] 0 1 0 0  
## [4,] 0 1 0 0  
## [5,] 0 0 1 0  
## [6,] 0 0 1 0  
## [7,] 0 0 0 1  
## [8,] 0 0 0 1

## Model building

model1 <- keras\_model\_sequential()  
model1 %>%  
 layer\_dense(units = 256, activation = 'relu', input\_shape = c(2352)) %>%  
 layer\_dense(units = 128, activation = 'relu') %>%  
 layer\_dense(units = 4, activation = 'softmax')  
  
summary(model1)

## Model: "sequential"  
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## Layer (type) Output Shape Param #   
## ================================================================================  
## dense\_2 (Dense) (None, 256) 602368   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## dense\_1 (Dense) (None, 128) 32896   
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
## dense (Dense) (None, 4) 516   
## ================================================================================  
## Total params: 635,780  
## Trainable params: 635,780  
## Non-trainable params: 0  
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#2352\*256  
#602112+256  
#128\*256  
#32768+128  
#128\*4  
#512+4  
#602368+32896+516 #=635780 total number of parameters

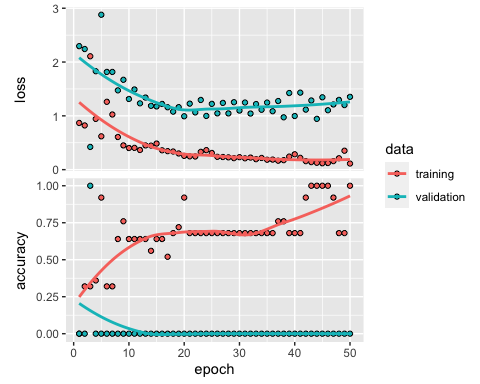
## Compile

model1 %>%  
 compile(loss = 'binary\_crossentropy',  
 optimizer = optimizer\_rmsprop(),  
 metrics = c('accuracy'))

## Fit the model

history <- model1 %>%  
 fit(x.train,  
 train.labels,  
 epochs = 50,  
 batch\_size = 52,  
 validation\_split = 0.2)  
  
plot(history)

## `geom\_smooth()` using formula 'y ~ x'



## Model Evaluation and Prediction - Train Data

model1 %>% evaluate(x.train, train.labels)

## loss accuracy   
## 0.3914616 0.7187500

pred <- model1 %>% predict\_classes(x.train)  
table(Predicted = pred, Actual = y.train)

## Actual  
## Predicted 0 1 2 3  
## 0 6 0 0 0  
## 1 0 8 0 0  
## 2 2 0 8 7  
## 3 0 0 0 1

prob <- model1 %>% predict\_proba(x.train)  
  
cbind(prob, Predicted = pred, Actual = y.train)

## Predicted Actual  
## [1,] 1.425269e-01 0.0015203404 0.8539401889 2.012557e-03 2 0  
## [2,] 3.318906e-01 0.0011318539 0.6655150056 1.462483e-03 2 0  
## [3,] 6.361971e-01 0.0030070601 0.3570580482 3.737714e-03 0 0  
## [4,] 7.188933e-01 0.0020321882 0.2763397098 2.734825e-03 0 0  
## [5,] 6.532916e-01 0.0025689385 0.3405857682 3.553701e-03 0 0  
## [6,] 8.037921e-01 0.0023923402 0.1908579320 2.957698e-03 0 0  
## [7,] 7.682139e-01 0.0034676176 0.2244487852 3.869768e-03 0 0  
## [8,] 8.110208e-01 0.0023823213 0.1833859831 3.210840e-03 0 0  
## [9,] 1.934965e-04 0.9977346659 0.0020218138 4.991511e-05 1 1  
## [10,] 5.231700e-04 0.9888200164 0.0105514722 1.053456e-04 1 1  
## [11,] 1.179871e-04 0.9988378882 0.0009932447 5.078492e-05 1 1  
## [12,] 2.405534e-05 0.9997892976 0.0001411741 4.541786e-05 1 1  
## [13,] 9.476185e-05 0.9991156459 0.0007292546 6.022683e-05 1 1  
## [14,] 8.130915e-05 0.9990758896 0.0007829532 5.976864e-05 1 1  
## [15,] 1.058037e-04 0.9986540079 0.0011684368 7.169516e-05 1 1  
## [16,] 7.066224e-05 0.9992114305 0.0006627170 5.510107e-05 1 1  
## [17,] 8.279671e-03 0.0002701973 0.9912852645 1.648553e-04 2 2  
## [18,] 5.577167e-03 0.0004100450 0.9937849045 2.279195e-04 2 2  
## [19,] 2.871355e-03 0.0002807697 0.9967141151 1.337326e-04 2 2  
## [20,] 3.274988e-03 0.0003147232 0.9962592125 1.511061e-04 2 2  
## [21,] 3.341396e-03 0.0007861264 0.9957002401 1.722211e-04 2 2  
## [22,] 4.349734e-03 0.0004623507 0.9950882792 9.959844e-05 2 2  
## [23,] 4.733725e-03 0.0005567275 0.9945695996 1.399600e-04 2 2  
## [24,] 4.135896e-03 0.0005625776 0.9951719642 1.295658e-04 2 2  
## [25,] 7.217956e-03 0.0054241465 0.0239145700 9.634434e-01 3 3  
## [26,] 1.115745e-01 0.2004091293 0.6779385209 1.007780e-02 2 3  
## [27,] 9.491935e-02 0.0530411117 0.8396114111 1.242812e-02 2 3  
## [28,] 7.248002e-02 0.0203884915 0.8986880779 8.443436e-03 2 3  
## [29,] 5.718434e-02 0.0146748051 0.9228358865 5.304952e-03 2 3  
## [30,] 2.879603e-02 0.0035522068 0.9645270109 3.124684e-03 2 3  
## [31,] 1.677502e-01 0.1840646863 0.5924490094 5.573612e-02 2 3  
## [32,] 1.516153e-02 0.0046105240 0.9788668752 1.361026e-03 2 3

## Evaluation and prediction on the test data

model1 %>% evaluate(x.test, test.labels)

## loss accuracy   
## 0.6953449 0.5000000

pred <- model1 %>% predict\_classes(x.test)  
table(Predicted = pred, Actual = y.test)

## Actual  
## Predicted 0 1 2 3  
## 1 0 2 0 0  
## 2 2 0 2 2