The Plant Pathology 2020 challenge dataset to classify foliar disease:

‘Diagnose apple plant diseases based on leaf images’

Rubino Massimiliano 800691

Abstract

Machine learning has emerged along with big data technologies and high-performance computing crating new opportunities for data intensive science in the multi-disciplinary agri-technologies domain. In this article I am going to explain why the use of the computer vison in agricultural field is important. In order to do this, I draw on a competition which goal was to diagnose plant disease solely based on leaf images. Solving this problem is important because diagnosing plant diseases early can save tons of agricultural products every year. This will benefit not only the general population by reducing hunger, but also the farmers by ensuring they get a greater harvest.

1. Introduction

The U.S. apple industry, annually worth $15 billion, experiences annual losses of millions of dollars due to various biotic and abiotic stresses, ongoing stress management, and multi-year impacts of the loss of fruit trees. Over the growing season, apples are under constant threat of a large number of insects, fungal, bacterial and viral pathogens, particularly in the Northeastern U.S [1]. Depending on the rate and severity of infection by diseases and insects, impacts range from unattractive cosmetic appearance, low marketability and poor quality of fruit, to decreased yield or complete loss of fruit or trees, causing huge economic losses [1].

In recent years, digital imaging and machine learning have shown great potential to accelerate plant disease diagnosis [2]. The digital imaging revolution has already created tremendous opportunities in many fields of social and professional life, and much of the world has ready access to a smartphone with a digital camera that can be used to capture symptoms of disease excellently. Computer vision methods are being developed to make use of digital images of symptoms of disease classification. This method merges human experience and machine learning to find visual patterns for grouping and

identification. The computer methods are now applicable, because in the past it was extremely difficult to collect enough high-quality real-life images to train the computer vision model.

Thanks to this technological revolution and the possibility to have this big data, the use of Deep Convolution Neural Network (CNN) models and other machine learning methods are getting increasingly important to classify diseased leaf images of crop plants. Recent publications [3] [4] have explored the use of computer vision to identify diseases in crops . A CNN based approach was used to detect and distinguish banana speckle disease from healthy leaves under difficult photographic conditions, complex backgrounds, different resolutions, several orientations, and various illuminations. However, all these published methods have limited capability in identifying a specific disease from many symptoms.

Therefore, this study has two main goals. First, overcoming the problems which could not be resolved in previous publications, using a more complete dataset concerning apple diseases. And second, explaining the potential use of CNN algorithm not merely as forecasting but also as explanatory device.

2. Methods

In this section I will explain how I got my prediction explaining the dataset and the employed model. The dataset is a collection of 3,651 images of apple foliar diseases and the images were captured in 2019 in Cornell AgriTech, Geneva, New York. All pictures are in RGB scale and vary considerably in their degree of illumination, angle, surface and noise in order to reflect real-life scenarios (Figure 1).

Figure 1 Sample of the pictures from the dataset. The pictures represent all 4 possible classes.

The classes that I have to detect are: healthy, multiple diseases, rust and scab. The categories are unbalanced, only 5% of the plants have multiple diseases, but healthy, rust and scab take up approximately one-third each.

Before training the CNN, I had to convert these pictures into comprehensible data to be ready for further processing. Thus, I converted all the pictures in tensor which dimensions are 150, 150, 3. The first two numbers refer to the height and length of the picture and the latter to the three colors RGB. This is a structure that the algorithm can understand.

Regarding the prediction model because the nature of the data and the research question, the use of CNN has worked well in terms of prediction and accuracy. Although, this type of model has a huge prediction power, it lacks significantly in terms of result interpretability. To overcome this problem, I found a way to explain the CNN with the help of the Lime library. The basic idea is to understand how a machine learning model, in this case deep neural network, predicts that an instance, for example an image, belongs to a certain class [5].

3. Results

The dataset has divided 3651 images into 70% for training and 30% for the validation-test. I used two different CNN models, the first one is a normal neural network with 3 million of parameters. Because of scarce results of this, the second model that I used was a pre-trained CNN the 'xception' with the technique of features extraction. I was able to get 97% of accuracy on the training data, and around 87% of accuracy on the validation-test (Figure 2).

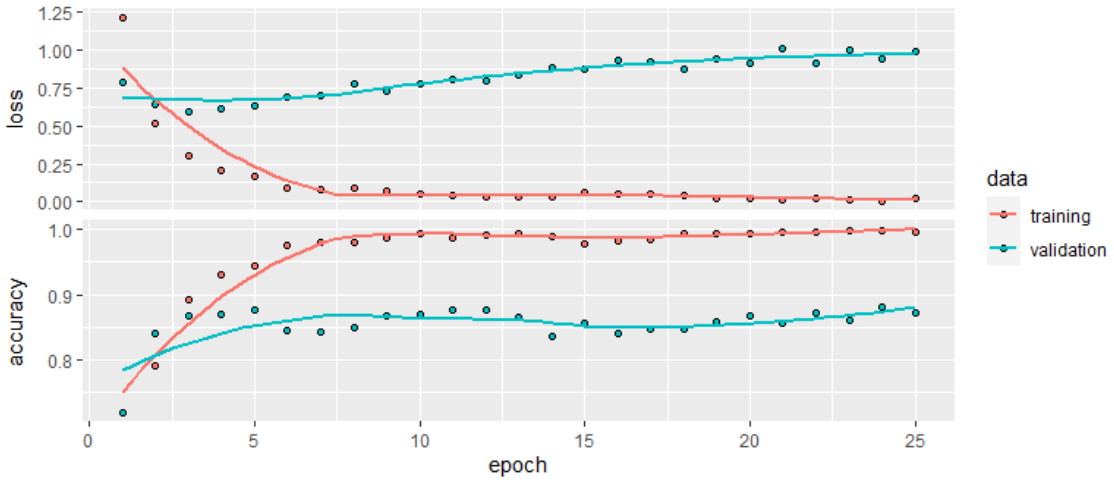

Figure 2 Performance in terms of accuracy and loss function. Model CNN xceprion.

Concerning the second goal of my project I used the library LIME available in Python. This is a modern Technique that tries to explain which are the most important features that the CNN has learnt. To do this I divided the pictures in super-pixels, for simplicity each super-pixel is one feature. Then I ran the model on it and in the end, I obtained the area of the image that contributed to the prediction. (Figure 3) depicts a prediction of rust disease.

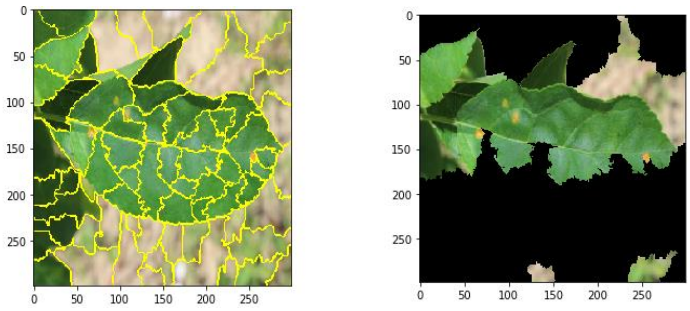

Figure 3 Super-pixels on the left, and area of the prediction on the right.

As we can see the model, it is able to capture the most part of the leaf. Thanks to this technique, we can start to use the CNN not only like a prediction model but also we can try to answer the question about interpretability of the results.

4.What economists can add to ML

When we use only a ML model, the risk of picking up spurious correlations and discovering relations that are not general is high. In order to overpass this problem, the use of an econometrics approach is necessary to understand why a model works and whether it has learned plausible relationships [8].

One of the central challenges facing Machine Learning is to combine data-driven methods with theoretical disciplines.

In this project I have developed an expert apple diseases detector. The use of this kind of tool in the agriculture field can be extremely profitable for all agents, clients as well as farmers. Regarding the farmers this can be advantageous in terms of saving money and in the use of fewer pesticides on their fields. Furthermore, for the clients this automatically translates in better pricing and quality.
Essentially, in this article I discussed how to answer a predictive question. However, if we really want to measure the impact of the new technology in the agriculture, what we need to use is an econometrics approach rather than a data science approach . In other words, we can frame a research question as either causal or predictive.

Before trying to answer a causal question, it is better to underline the main characteristics of a causal model:

- It explains the causation rather the association.
- The use of a theory is necessary before estimating the model rather than a data driven approach.
- Testing a set of hypotheses is more important than predict new observations.
- It focuses on minimizing the bias to get the correct impact of x on y rather than control the bias-variance trade off [6].

As we can see, in a causal study the main goal it is to explain rather than predict, and because of this it is also crucial to evaluate the power of the relationship between x and y. To do this it is important to correctly compute the standard error and properly evaluate the statistical significance of the individual variables. After discussing the main differences between causal and predictive question, now we can try to answer a potential causal question, an interesting question could be about the role of the technology in the agriculture.

Because we already developed a useful tool to detect diseases of apple leaves, it can be extremely interesting to evaluate the economic impact of the use of this technology on the harvest income.

The causal question that could be asked is the following:" Does the disease detector impact the harvest? And if yes, how?".

To answer the following question one could set up a random experiment using Rubin's Causal Model [7]. Thanks to this model it is possible to solve the selection problem and use a simple differences estimator to evaluate the significance of the use of the detector on the harvest income.

5. Conclusion

In summary the use of the computer vison has demonstrated great potential in prediction problem. As noted in the section 4 applying causal analysis to ML methods can be used to evaluate the impact of the disease detector on the harvest.

I conclude by highlighting the latest technological development that ML is particularly relevant for agricultural and applied economics. Therefore, employing ML methods to causal analysis is a new and growing field.

Thus, merging ML to traditional identification methods is not only becoming a new field in science but also provides a basis for an agricultural revolution.

References

[1] N. S. S. B. A. K. Ranjita Thapa, "The Plant Pathology 2020 challenge dataset to classify foliar disease of apples," Cornell Tech,, NY 10044, USA, 2020.

[2] A.-K. Mahlein, "Plant disease detection by imaging sensors-parallels ad specific demands for precision agriculture and plant phenotyping," *Plant disease,* vol. 100, no. 2, pp. 241-251, 2016.

[3] J. Barbedo, "An automatic method to detect and measure leaf disease symptoms using digital image processing," *Plant Disease*, vol. 98, no. 14, pp. 1709-1716, 2014.

[4] e. a. Y. Atoum, "On developing and enhancing plant-level disease rating systems in real fields," *Pattern Recognition*, vol. 53, pp. 287-299, 2016.

[5] M. T. R. a. S. S. a. C. Guestrin, "Why Should I Trust You?": Explaining the Predictions of Any Classifier," 2016.

[6] G. Shmueli, "To Explain or to Predict?," *Statistical Science,* vol. 23, no. 3, pp. 289-310, 2010.

[7] D. B. RUBIN, "ESTIMATING CAUSAL EFFECTS OF TREATMENTS IN RANDOMIZED AND NONRANDOMIZED STUDIES," *Journal of Educational Psychology*, vol. 66, no. 5, pp. 688-701, 1974.

[8] K. B. a. T. H. Hugo Storm, "Machine learning in agricultural and appliesd economics," *European Review of Agricultural Economics,* p. 1–44, 2019.

R Code

```
library(keras)

train_dir <- "C:/Users/Massimiliano/Desktop/MAGISTRALE/secondo anno/economics f
or data scienze/Report/foglie/train"
img_dir<-"C:/Users/Massimiliano/Desktop/MAGISTRALE/secondo anno/economics for d
ata scienze/Report/images"
library(readr)
train <- read_csv("C:/Users/Massimiliano/Desktop/MAGISTRALE/secondo anno/econom
ics for data scienze/Report/train.csv")
#converto lable in etichetta y
y<-numeric()
for (i in 1:dim(train)[[1]]) {
  y[i]<-colnames(train)[which(train[i,]==1)]
}
#salvo posizioni immagini train
H<-which(y=='healthy')-1
M<-which(y=="multiple_diseases")-1
R<-which(y=="rust")-1
S<-which(y=="scab")-1
```

```r
#creo cartelle dati
train_healthyr <- file.path(train_dir, "healthy")
dir.create(train_healthyr)
train_multiple_diseases <- file.path(train_dir, "multiple_diseases")
dir.create(train_multiple_diseases)
train_rust <- file.path(train_dir, "rust")
dir.create(train_rust)
train_scab<- file.path(train_dir, "scab")
dir.create(train_scab)
fnames <- list.files(train_dir, full.names = TRUE)



#copia immagini in directory helty
fnames <- paste0("Train_",H, ".jpg")
file.copy(file.path(img_dir, fnames),file.path(train_healthyr))

fnames <- paste0("Train_",S, ".jpg")
file.copy(file.path(img_dir, fnames),
          file.path(train_scab))

fnames <- paste0("Train_",M, ".jpg")
file.copy(file.path(img_dir, fnames),
file.path(train_multiple_diseases))

fnames <- paste0("Train_",R, ".jpg")
file.copy(file.path(img_dir, fnames),
file.path(train_rust))


train_datagen <- image_data_generator(rescale = 1/255)

train_generator <- flow_images_from_directory(
  # target directory
  train_dir,
  # target data generator
  train_datagen,
  # Tutte le immagini sono riscalate tra 150 x 150
  target_size = c(150, 150),
  batch_size = 15,
  # Utilizziamo questo tipo di classe perché necessitiamo di etichette binarie
)



model <- keras_model_sequential() %>%
  layer_conv_2d(filters = 128, kernel_size = c(3, 3), activation = "relu",
                input_shape = c(130, 200, 3)) %>%
  layer_max_pooling_2d(pool_size = c(3, 3)) %>%
  layer_conv_2d(filters = 128, kernel_size = c(3, 3), activation = "relu") %>%
layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 128, kernel_size = c(3, 3), activation = "relu") %>%
```

```r
layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 128, kernel_size = c(3, 3), activation = "relu") %>%
layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_flatten() %>%
  layer_dense(units = 512, activation = "relu") %>%
  layer_dense(units = 4, activation = "softmax")
summary(model)

model %>% compile(
  optimizer = 'adam',
  loss = "categorical_crossentropy",
  metrics = c('accuracy')
)


history <- model %>% fit_generator(train_generator,
  epochs = 15,
  steps_per_epoch = 18,validation_steps = 10)

#modello pre allenato-----
conv_base <- application_xception(
  # Pesi da utilizzare per iniziallizzare la rete
  weights = "imagenet",
  # Questo comando è per includere o meno il layere fully connected
  include_top = FALSE,
  # Dimensione del tensore che utilizzeremo come input della nostra rete
  input_shape = c(150,150, 3)
)

conv_base


datagen <- image_data_generator(rescale = 1/255)
batch_size <- 20


extract_features <- function(directory, sample_count) {

  features <- array(0, dim = c(sample_count, 5, 5, 2048))
  labels <- array(0, dim = c(sample_count,4))

  generator <- flow_images_from_directory(
    directory = directory,
    generator = datagen,
    target_size = c(150, 150),
    batch_size = batch_size
  )

  i <- 0
  while(TRUE) {
    batch <- generator_next(generator)
    inputs_batch <- batch[[1]]
    labels_batch <- batch[[2]]
    features_batch <- conv_base %>% predict(inputs_batch)
    index_range <- ((i * batch_size)+1):((i + 1) * batch_size)
```

```
    features[index_range,,,] <- features_batch
    labels[index_range,] <- labels_batch

    i <- i + 1
    # Per bloccare il ciclo
    if (i * batch_size >= sample_count)
      break
  }

  list(
    features = features,
    labels = labels
  )
}

train <- extract_features(train_dir, 1800)

reshape_features <- function(features) {
  array_reshape(features, dim = c(nrow(features),  5 * 5 * 2048))
}
train$features <- reshape_features(train$features)


model <- keras_model_sequential() %>%
  layer_dense(units = 512, activation = "relu",
              input_shape = 5 * 5 * 2048) %>%
  layer_dense(units = 128,activation = 'relu')%>%
  layer_dropout(rate = 0.2)%>%
  layer_dense(units = 4, activation = "softmax")

model %>% compile(
  optimizer = 'adam',
  loss = "categorical_crossentropy",
  metrics = c('accuracy')
)


history <- model %>% fit(
  train$features, train$labels,
  epochs = 25,
  batch_size = 200,validation_split=0.30
)

plot(history)
```