Plant Disease Classification Using Convolutional Networks and Generative Adverserial Networks

Emanuel Cortes
Stanford University
ecortes@stanford.edu

Abstract

Crop diseases serve as a major threat to food supply. As a result of the growing of smartphone technology throughout the world, it has now become technical feasible to leverage image processing techniques to identity type of plant disease from a simple photo. Identifying disease can lead to quicker interventions that can be implemented to reduce the effects of crop diseases on food supply.

Using a public dataset of 86,147 images of diseased and healthy plants, a deep convolutional network and semi supervised methods are trained to classify crop species and disease status of 57 different classes.

1. Introduction

Despite having seen many improvements in the mass production and accessibility of food, food security remains threatened by a variety of factors such as the decline of pollinators and plant diseases. In the developing world, more than 80 percent of the agricultural production is generated by smallholder farmers, and reports of yield loss of more than 50% due to pests and diseases are common. Furthermore, the majority of individuals suffering from hunger live in smallholder farming households. Fortunately, diseases can be managed by identifying the diseases as soon as it appears on the plant. In addition, with the rise of the internet and mobile technology worldwide, it easy to access diagnosis information on a particular type of disease. As a result, the prevalence of smartphones with powerful cameras can help to scale up any type of solution that involves crop detection feasible and practical.

Smartphones in particular offer very novel approaches to help identify diseases because of their computing power, high-resolution displays, and extensive built-in sets of accessories, such as advanced HD cameras. In fact it is estimated that around 6 billion phones would be available around 2050.

The input to the algorithm in this paper will 2D images of diseased and healthy plant leaves. I will be using a deep convolutional network, a generative adversarial network, and a semi supervised learning approach that utilizes a ladder network. These different approaches will be used to output a predicted disease type or a type of healthy plant species.

2. Related Work

2.1. Plant Disease Classificaiton

Before the problem of crop disease detection can be solved, the problem of identifying different species of plants need to be addressed. Fortunately, there has been much work already completed in this problem domain. In the research article, A Model of Plant Identification System Using GLCM, Lacunarity, and Shen Features, researchers dived into many preprocessing steps that can be done to extract important features for binary-class classification. Images are transformed using Polar Fourier Transform to achieve translational and rotational invariance. Color features, such as the mean, standard deviation, skewness, and kurtosis are made on the pixel values of the leaves. Lastly, the research paper incorporated features that come from gray-level co-occurrence matrix (GLCM). The GLCM functions characterize the texture of an image by calculating how often pairs of pixel with specific values and in a specified spatial relationship occur in an image, creating a GLCM, and then extracting statistical measures from this matrix. Although crop disease detection is moving away from hand-engineered features, such as SIFT, HoG, and SURF, there are still a variety of techniques to help deep learning networks extract important features, such as background segmentation. The method that was used in the experiments will be explained in the method section.

In the research paper, Plant Leaf and Disease Detection by Using HSV Features and SVM, the researchers proposed using a neural network to classify whether a leaf was infected or not. If a leaf was infected, the images were further processed by a neural network, where a genetic algorithm was implemented to optimize the SVM loss to determine

identify the type of disease. This method is quite interesting in that it breaks down the process disease identification in two steps. It will be interesting to compare and contrast with more recent papers, where healthy leaves are treated as just another class label. As a result, classification is just done in a single step. In addition, the research paper introduces a method for optimizing the loss function using a genetic algorithm, which can be compared to natural selection where only the strong hyper parameters will survive. I need to do more research in how genetic algorithms compare in other forms of computing loss, such as Adam, RmsProp, and etc. Lastly, similar to the paper, A Model of Plant Identification System Using GLCM, Lacunarity, and Shen Features, there was a focus on identifying important features of the image to help in the classification process. In both papers, they used gray-level co-occurrence matrix (GLCM) to extract information about the distribution of pixel values.

2.2. Plant Disease Classification with Convolutions

Other work moved towards using convolutional networks to get better performances. Convolutional Networks are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. ConNets have been successful in identifying faces, objects, and traffic signs apart from powering vision in robots and self driving cars.

ConvNets derive their name from the "convolution" operator. The primary purpose of Convolution in case of a ConvNet is to extract features from the input image. Convolution preserves the spatial relationship between pixels by learning image features using small filters.

There are many important parts of the Convolution Network. These includes the following properties

1. Depth: Depth corresponds to the number of filters we use for the conolution operation.

2. Stride: Stride is the number of pixels by which we slide our filter matrix over the input matrix.

3. Zero-padding: Sometimes, it is convenient to pad the input matrix with zeros around the border, so that we can apply the filter to bordering elements of our input image matrix.

4. Non-Linearity: ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero. The purpose of ReLU is to introduce non-linearity in our ConvNet, since most of the real-world data we would want our ConvNet to learn would be non-linear (Convolution is a linear operation element wise matrix multiplication and addition, so we account for non-linearity by introducing a non-linear function like ReLU).

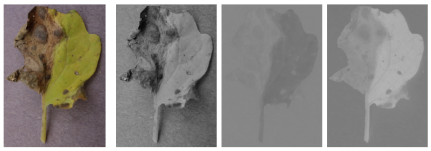
5. Spatial Pooling: Spatial Pooling (also called subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: Max, Average, Sum etc. These networks have grown in the number of layers leading to architectures such as RestNet and AlexNet that have been trained on images such as Cifar-10 and then fined tune to other problems, such as plant classification.

3. Method

3.1. Background Segmentation

In the research paper, "Using Deep Learning for Image-Based Plant Disease Detection," Mohanty and his colleagues worked with three different versions of the leaf images from PlantVillage. One of these versions included leaf images that were segmented to to exclude the background. According to Mohanty and his colleagues, these segmented images of the whole dataset was prepared to investigate the role of the background on the learning algorithm.

In order to segment the background, I would be using a model that is inspired by the Boris Conforty tutorial on segmenting images from PlantVillage. If you take a picture of a uniform background under non-uniform lightning conditions, you'll end up with a gradient in all R, G and B channels (in the RGB color space), whereas only the L channel will be affected (in the Lab color space). Here is an example image expressed in the channels of the lab colore space:



Because the negative values in the a channel corresponds to green pixels, I created a mask that is able to zero out the pixel locations of the original picture that is located outside the region where the values in the lab space are negative. In addition, I blurred the image with a guassian to ensure that the negative values that were chosen truly belong to the area that represented the leaf of the original leaf. Here is the following example of the image:

3.2. Discriminator as a Classifier

The discriminator of a generative adversarial network is used to discriminate the fake images from real images. We can further generalize the discriminator to be a classifier that with two classes: FAKE and REAL. If this is the case, then it may not be too hard to write a discriminator that operatons

on more than two classes. As part of my attempt to design a semi-supervised approach, I implemented a loss function that could create a discriminator that could act as a classifier for different plant diseases.

Before explaining my multi-label categorical generative network, previous work has been done that influenced how I constuctured my model. In "Semi-Supervised Learning with Generative Adverserial Networks," Augustus Odena proposed that the descirminator network D can be implemented with a softmax output layer with one unit for each of the classes [REAL, FAKE]. Once this modification is made, its simple to see that D could have N+1 output units corresponding to CLASS-1, CLASS-2, . . . CLASS-N, FAKE]. Therefore the discriminator can act as a classifier. In both my research paper and in Odena's work, this type of discrimnatory is referred to as D/C.

Odena further describes that training this type of GAN is quite similar to training a normal GAN. D/C is trained to minimize the negative log likelihood with respect to the given labels and G is trained to maximize it. Therefore, the following algorithm is proposed to train my algorithm

Algorithm 1 SGAN Training Algorithm

Input: I : number of total iterations
for $i > 1$ **to** $i == 10$ **do**
    Draw m noise samples
    Draw m examples from the real data
    Perform gradient descent to minimize the D/C loss
    function with respect to real and fake lables
    Draw m noise samples
    Perform gradient descent on the parameters of G w.r.t.
    the NLL of D/Cs outputs on the minibatch of size m.

In addition, here is the following code for my loss function for the descriminator:

```
def discriminator_loss_for_mcgan
(logits_real, logits_fake,
true_labels,
 softmax_loss):

Inputs:
- logits_real:
PyTorch Variable of shape (N, C)
giving scores for the real data.
- logits_fake:
PyTorch Variable of shape (N, C)
giving scores for the fake data.
- true_labels:
PyTorch Variable of Shape (N,)
giving labels for real data.
loss = None
N, C = logits_real.size()
fake_labels =
Variable((torch.zeros(N)+23).type(dtype).long())
return softmax_loss(logits_real, true_labels) +
          softmax_loss(logits_fake, fake_labels)
```

The discriminator loss function will contain the sum of two softmax functions. One of softmax will try to reduce the negative log likilhood with respect to the given labels of the real data. The other softmax function will reduce the negative log likilhood with respect the fake labels of the fake data. A loss function that reduces the sum of these softmax functions would create a discriminator that discriminate between real and fake data and if given an real image, the discriminator can produce a label or identify a particular type of disease for that image.

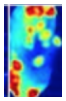
4. Dataset and Features

There is a total of 86,147 images of diseased and healthy plants. These images span 25 different species of plants. Each set of images, such as the training, validation, and testing, span 25 different species of plants. The training set includes 55,135 images of disease and healthy plant leaves. that also span 25 different species of plants. The validation set includes 13,783 images of disease and healthy plant leaves. The testing set includes 17, 229 images of disease and healthy plant leaves.

All images were segmented to remove the background. In addition, all images were resized to have a width of 64 pixels and a height of 64 pixels. Then this was repeated for images that were segmented.

5. Experiments/Results

The experiment that performed well on the unsupervised data was rsnet. It was able to score above 80% in the training phrase under 5 epochs with the learning rate of 1e-5. This was our baseline and was expected to do well. Most of the activiations were completey dark after finetuning the the data. The following images are part of an activation layer that shows that resnet is learning to identify diseases in few of the activations.



There were two experiments that were done for implementing DCGAN. In one case, I used the following architecture for the descriminator:

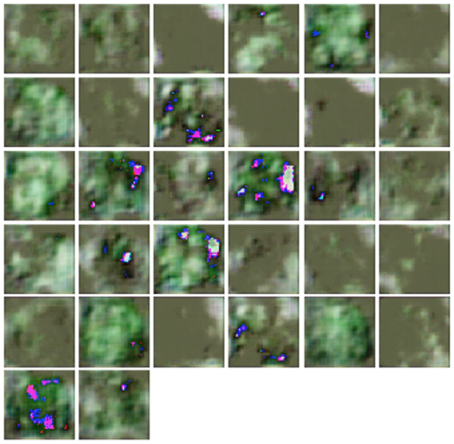
```
Unflatten(BATCH_SIZE, NUM_CHANNELS, I
MAGE_HEIGHT, IMAGE_WIDTH),
nn.Conv2d(NUM_CHANNELS, 32,
kernel_size=5, stride=1),
nn.LeakyReLU(0.01, inplace=True),
nn.MaxPool2d(kernel_size=2, stride=2),
nn.Conv2d(32, 64, kernel_size=5, stride=1),
nn.LeakyReLU(0.01, inplace=True),
nn.MaxPool2d(kernel_size=2, stride=2),
Flatten(),
nn.Linear(10816, 4*4*64),
nn.LeakyReLU(0.01, inplace=True),
nn.Linear(4*4*64, NUM_CLASSES)
```

, where batch size is 64, num channels is 3, image height is 64, image width is 64, and num of classes is 37. This model lachieved an accuracy of 78% for testing data. However, it performed poorly on the unstructured images, which only performed an acurracy of 6%.

I used the following architecture for the generator:

```
nn.Linear(noise_dim, 1024),
nn.BatchNorm1d(1024),
nn.Linear(1024, 16*16*128),
nn.BatchNorm1d(16*16*128),
Unflatten(BATCH_SIZE, 128, 16, 16),
torch.nn.ConvTranspose2d(128, 64,
kernel_size=4, stride=2, padding=1),
nn.ReLU(inplace=True),
nn.BatchNorm2d(64),
torch.nn.ConvTranspose2d(64, 3,
kernel_size=4, stride=2, padding=1),
nn.Tanh(),
Flatten()
```

In addition here is the first image produced the early iterations during training:



And this the last image produced during training.

In order to improve the quality of images produced by generative model and increase the accuracy of the unstructured data, I increased the complexity of both the descriminator and generator. Here is the model for the descriminator:

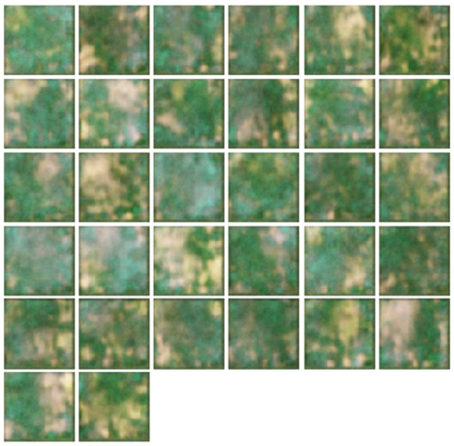
```
Unflatten(BATCH_SIZE, NUM_CHANNELS,
IMAGE_HEIGHT, IMAGE_WIDTH),
nn.Conv2d(NUM_CHANNELS,
128, kernel_size=5, stride=1),
nn.LeakyReLU(0.01, inplace=True),
nn.MaxPool2d(kernel_size=2, stride=2),
nn.Conv2d(128, 64, kernel_size=5, stride=1),
nn.LeakyReLU(0.01, inplace=True),
nn.MaxPool2d(kernel_size=2, stride=2),
nn.Conv2d(64, 32, kernel_size=5, stride=1),
nn.LeakyReLU(0.01, inplace=True),
nn.MaxPool2d(kernel_size=2, stride=2),
Flatten(),
nn.Linear(32*4*4, 4*4*64),
nn.LeakyReLU(0.01, inplace=True),
nn.Linear(4*4*64, NUM_CLASSES)
```

Here is the model for the descriminator:

```
 nn.Linear(noise_dim, 1024),
nn.BatchNorm1d(1024),
nn.Linear(1024, 32*32*128),
nn.BatchNorm1d(32*32*128),
Unflatten(BATCH_SIZE, 128, 32, 32),
torch.nn.ConvTranspose2d(128, 64,
kernel_size=4, stride=2, padding=1),
nn.ReLU(inplace=True),
torch.nn.ConvTranspose2d(64, 32,
kernel_size=4, stride=2, padding=1),
nn.ReLU(inplace=True),
torch.nn.ConvTranspose2d(32, 16,
kernel_size=1, stride=1, padding=0),
```

```
nn.Conv2d(16, 3, kernel_size=2, stride=1),
nn.LeakyReLU(0.01, inplace=True),
nn.MaxPool2d(kernel_size=1, stride=2),
nn.Tanh(),
Flatten()
```

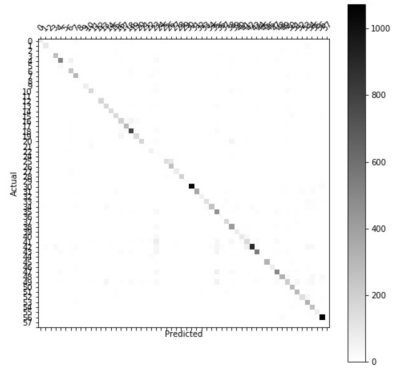
Here is the improved image at the beginning of training:



Here is the final improved image at the beginning of training:

In addition, the classifier performed worst in classifying the test data (69%) However, the performance in the unstructured data was improved to 12%

Here is the confusion matrix for the test data:



6. Conclusion

Using Gans may hold promise to classify diseases in classifying plant disease. Unfortunately segmenting based on background did not improve accuracy. One possible idea for future work is to implement segmentation with Mask RCNN.

1.) Strange RN, Scott PR (2005) Plant disease: a threat to global food security. Phytopathology 43

2.) Dirzo, Rodolfo; Hillary S. Young; Mauro Galetti; Gerardo Ceballos; Nick J. B. Isaac; Ben Collen (2014).”Defaunation in the Anthropocene”(PDF).Science.345(6195): 401406. 3.) UNEP (2013) Smallholders, food security, and the environment.

4.) Harvey CA et al. (2014) Extreme vulnerability of smallholder farmers to agricultural risks andclimate change in madagascar. Philosophical Transactions of the Royal Society of London B:Biological Sciences 369(1639). 5.) Mohanty, S. P., Hughes, D. P., Salath, M. (2016). Using Deep Learning for Image-Based Plant Disease Detection. Frontiers in Plant Science, 7. doi:10.3389/fpls.2016.01419 6.) Hanson, A.J., Joy, A., Francis, J. Francis, J. (2017). Plant Leaf Disease Detection using Deep Learning and Convolutional Neural Network. IJESC, Volume 7 Issue No.3 . 7.) Sladojevic, S, Arsenovic, M., Anderla, A., Culibrk, Dubravko, Sefanovic, D. (2016). Deep Neural Networks Based Recognition of Plant Disease by Leaf Image Classification. 8.) Naik, M.R., Sivappagari, C. (2016). Plant Leaf and Disease Detection by Using HSV Features and SVM. IJESC, Volume 6 Issue No.12. 9.) Kadir, A. (2014). A Model of Plant Identification System Using GLCM, Lacunarity and Shen Features. Research Journal of Pharmaceutical, Biological, and Chemical Sciences Vol.5(2)