Plant Disease Detection

Sevda SAYAN
Hacettepe University
21527305
b21527305@cs.hacettepe.edu.tr

Fatmanur TURHAN
Hacettepe University
21527429
b21527429@cs.hacettepe.edu.tr

Ismet SEYHAN
Hacettepe University
21693431
b21693431@cs.hacettepe.edu.tr

January 15, 2020

Abstract

In crops produced as a result of agricultural activities, diseases caused by some environmental factors or genetic structure of the plant prevent the productive production. Our main objective in this project is to enable people interested in agriculture to accurately and practically detect diseases in their crops and to implement the most accurate treatment. (1) For this purpose, we have set a target as future work in our project and we will try to make this work a project that can be used on mobile platform.

1 Introduction

Various methods are available to determine if a plant has a disease. One of these methods is to look at the leaves of plants. Discoloration on the leaf, yellowing, fading, desiccation, regional staining, the formation of intermediate holes and such as these symptoms is a sign of disease. Each plant type has its own specific diseases. For example; black rot disease in apple, powdery mildew disease in cherry, common rust disease in corn, leaf blight disease in grapes, etc.

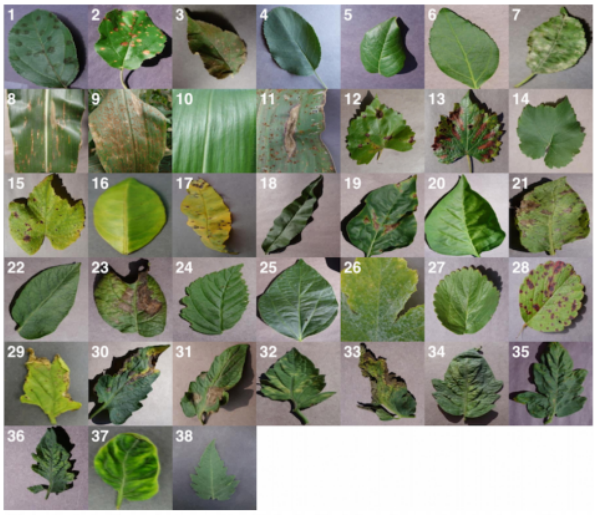
We found two datasets (2) suitable for our project and we combined two datasets so our own dataset is created. This resulted in a larger dataset and more classes. So the functionality of our project is increased. In each class of the dataset different numbers of images exist. Some of them contained 4000 images. We decided to reduce the dataset to make the transactions easier. It also included photos for the current dataset train and test. So we decided to produce an unseen validation part of the dataset from the test part of the dataset.

Our dataset contains both healthy and diseased plant leaf samples for each plant. There are usually 3 or 4 disease classes for each plant. In total, consists of 38 classes. There are 300 train images, 50 validation images and 50 test images for each class. Our dataset plants: tomato, squash, raspberry, potato, grape, strawberry, pepper, peach, orange, corn, cherry, soybean, blueberry, apple. Figure 1 contains a sample image for each class of our dataset. When starting the project, articles related to the projects using the same dataset were searched. Related Work section has some information about this article.

2 Related Work

As a result of the literature survey, articles of similar projects were found.In our reference article (3), "Plant Village" dataset containing approximately 54 thousand plant leaf images was examined.Each class label is a pair that crops and diseases. Estimates plant disease from a

given image of the plant.In all approaches studied in this article, images up to 256 pixels in height and width are resized.



(1) Apple Scab, Venturia inaequalis (2) Apple Black Rot, Botryosphaeria obtusa (3) Apple Cedar Rust, Gymnosporangium juniperi-virginianae (4) Apple healthy (5) Blueberry healthy (6) Cherry healthy (7) Cherry Powdery Mildew, Podoshaera clandestine (8) Corn Gray Leaf Spot, Cercospora zeae-maydis (9) Corn Common Rust, Puccinia sorghi (10) Corn healthy (11) Corn Northern Leaf Blight, Exserohilum turcicum (12) Grape Black Rot, Guignardia bidwellii, (13) Grape Black Measles (Esca), Phaeomoniella aleophilum, Phaeomoniella chlamydospora (14) Grape Healthy (15) Grape Leaf Blight, Pseudocercospora vitis (16) Orange Huanglongbing (Citrus Greening), Candidatus Liberibacter spp. (17) Peach Bacterial Spot, Xanthomonas campestris (18) Peach healthy (19) Bell Pepper Bacterial Spot, Xanthomonas campestris (20) Bell Pepper healthy (21) Potato Early Blight, Alternaria solani (22) Potato healthy (23) Potato Late Blight, Phytophthora infestans (24) Raspberry healthy (25) Soybean healthy (26) Squash Powdery Mildew, Erysiphe cichoracearum (27) Strawberry Healthy (28) Strawberry Leaf Scorch, Diplocarpon earlianum (29) Tomato Bacterial Spot, Xanthomonas campestris pv. vesicatoria (30) Tomato Early Blight, Alternaria solani (31) Tomato Late Blight, Phytophthora infestans (32) Tomato Leaf Mold, Passalora fulva (33) Tomato Septoria Leaf Spot, Septoria lycopersici (34) Tomato Two Spotted Spider Mite, Tetranychus urticae (35) Tomato Target Spot, Corynespora cassiicola (36) Tomato Mosaic Virus (37) Tomato Yellow Leaf Curl Virus (38) Tomato healthy.

Figure 1: Samples of each class categories

In the reference study, three distinct types were tested for the entire data set: the original type of PlantVillage data set, the gray type of PlantVillage data set, and the leaf segmentation type. All background details are removed if the leaves are divided into sections.

Figure 2 shows sample images from three tried types of the Plant Village dataset. This series of experiments was to understand whether the neural network really learned the “concept of plant diseases”, or the possibility of simply learning the innate trends in the dataset. According to our reference article, the best results in the classification were obtained by AlexNet architecture and GoogleNet architectures.

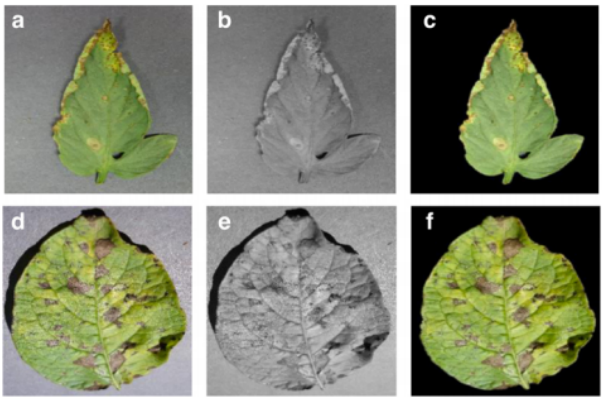


Figure 2: Three types of Plant Village dataset that tried in the reference paper. Shows that in the left column is an original type of dataset, in the middle column is a gray type of dataset and in the right column segmented type of dataset

3 Approach

Usefulness of deep convolutional neural networks is used for our classification. When talk about deep neural networks the first thing that comes to our mind is the convolutional neural networks. We focused on VGG16 and ResNet50 architectures in this project.

VGG16 proposed by two people who is from University of Oxford named. They submitted it to ILSVRC-2014 and published this prominent model. It has some reforms compare to AlexNet and became commonly used image classification model. The 16 in its name refers to it has sixteen layers and and it has approximately 138m parameters.

ResNet50 is another convolutional network that has really important advancements on some image classification tasks. It has 50 residual layer as can be understood by its name. When we go deep into the neural networks at one point they starts to getting worse. These type of situations can be solved by using residual layers.

Purpose of performing different types of configurations is reaching most accurate model on our project. To do this some different experiments are made.

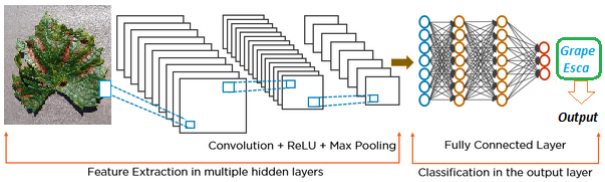


Figure 3: Steps of our network

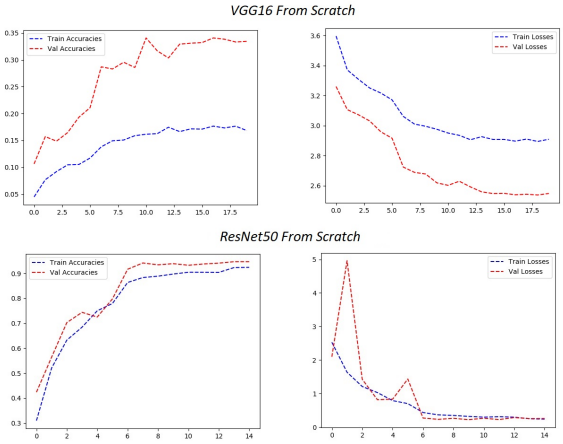


Figure 4: Train and validation accuracies/losses when training from scratch

4 Experimental Results

We have decided to try some experimental configurations by changing the following parameters:

Architecture

- VGG16
- Resnet50

Training mechanism

- Transfer Learning
- Training from Scratch

As a beginning, we have trained our VGG16 and ResNet50 model from scratch by using our belittled dataset. But gotten losses and accuracies was not satisfying on VGG16. As seen upper graphs above, train accuracy start with 4% and end with 16% when validation accuracy start with 10% and end with 33%. On the other hand ResNet50 train accuracy start with 31% and end with 92% when validation accuracy start with 42% and end with 94%.

Then we have trained our archictures using transfer learning training mechanism. Two models that we used have been trained on Imagenet and for using transfer learning weights we are re-initialized fc layer according to our number of class. As a result we obtained higher scores on both train and validation phases. On VGG16, train accuracy start with 54% and end with 85% when validation accuracy start with 81% and end with 94%. Besides, on ResNet50 train accuracy start with 69% and end with 97% when validation accuracy start with 98% and end with 76%.

In this experiments we used Adam(4) optimizer and learning rate is setted to 0.001. Each experiments runs for a total of 20 epochs, where epoch refers to one cycle through the full training dataset. On each epoch we usually observed decreasing losses and increasing accuracies as it should be.

Among the transfer learning and training from scratch mechanisms, transfer learning consistently performs better than training from scratch and always yields better results which were expected. After getting results in our first architecture, we decided to move on with ResNet50. However before choosing best model, we decided to observe performances of these models on unseen data. Losses and accuracies of those models on evaluation are shown below.

Test data set consists of nearly 2K samples and each class has fifty or more samples in itself. These samples are similar to train samples but are not exact copy of them. Similar to train and validation results, models behaved alike on test data. As seen in the Figure5, the best results are achieved on ResNet50 model that is trained us-

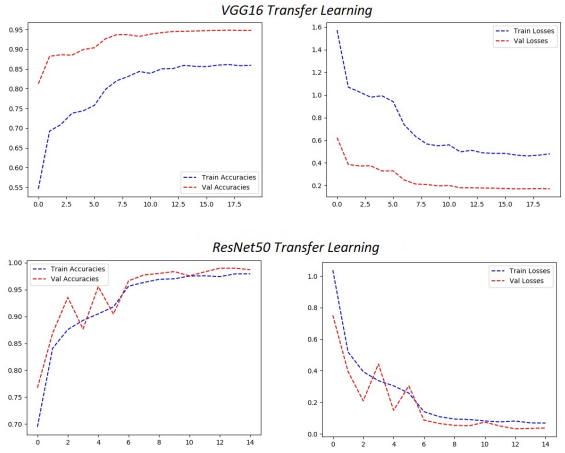


Figure 5: Train and validation accuracies/losses when transfer learning

	Loss	Accuracy
VGG16 From Scratch	0.0416	0.3209
VGG16 Transfer Learning	0.0030	0.9342
ResNet50 From Scratch	0.0027	0.9461
ResNet50 Transfer Leaning	0.0005	0.9887

Figure 6: Evaluation Results

ing transfer learning method. Resnets are called Residual Networks and operating logics are based on CNNs .

When we compare Resnets with VGG16, they are more deeply than VGG, and has 50 residual layers. Resnet advertised that has residual relationships among layers. Otherwise, layers of Resnet uses Batch Normalization, which has been integrated to VGG. Our final ResNet50 model predicted disease of plants true at the ratio of 98%.

When we focus on these visual results (Figure 11) we can clearly say that model can classify them properly even

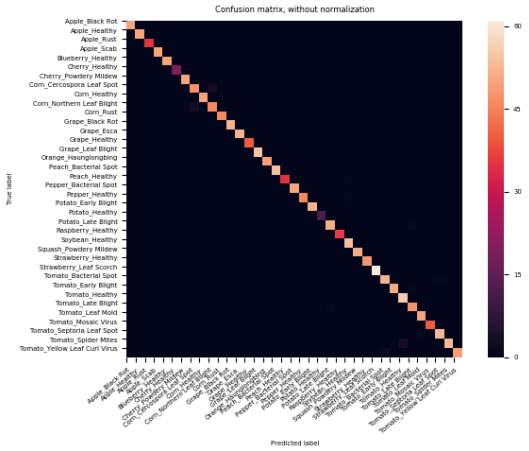


Figure 7: Confusion Matrix

on unseen data. Fifty percent of our classes have classified with accuracy of %100. Besides this, just %71.2 of Potato Late Blight classified correctly which has the lowest accuracy.

Assume we want to detect disease of input from an sample of a Tomato Early Blight. Weights of our neural network accountable for detecting disease of Tomato Early Blights should have a high weights because its values are tend to adjust their pixels to be high in Tomato Early Blights. These weight values can be interpreted as forming pattern of our classes. We are not saying our network anything about what these leafs are but they evoke us our classes anyway in a kind of interesting way (Figure 7).

5 Conclusion

We trained a model about diseases on plant leafs with advantages of deep convolutional neural network. This study is very important for farmers because they can find out disease of their plants or without necessity of feature engineers. Whole training operation has been took nearly two hours for every model on both transfer learning and training from scratch. Training time is a bit much even it executed on high performance server however in contrast final model's classification respond time is very low.

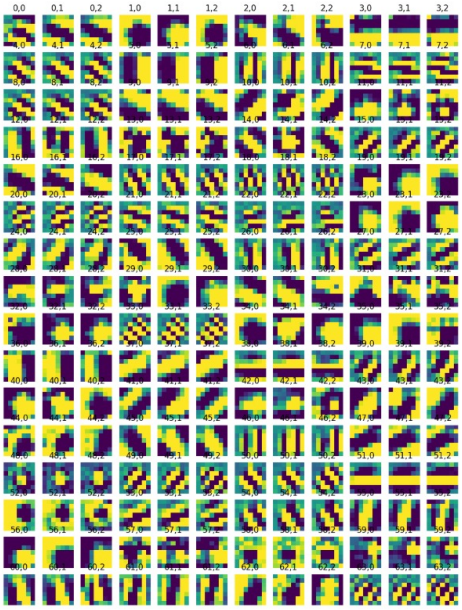


Figure 8: Final ResNet50 Model Weights

In the end inside our minimised PlantVillage data set of about 14K images including 38 classes and 26 diseases and health, our goal is accomplished with 98.8% accuracy for these diseases.

Figure 9: Strength of our model

Final selected model is durable even for very similar leafs, in other words among two classes that belongs to same plant but have different diseases model can distinguish these two types of classes.

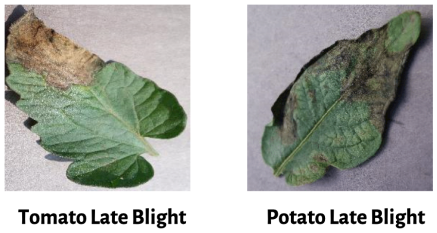


Figure 10: Weakness of our model

On the other hand final model can predict disease true but confuse about plant type as a weakness. In other words, among two classes that belongs to different plant bu have same diseases model sometimes can not distinguish these two types of disease types.

6 Future Work

Our next goal is to adapt chosen best model to the mobile platform. We will focus on the correct classification of photos selected from the sd card on a Android device or taken instantly via camera. We will end up integrating our system into the Android environment using PyTorch mobile. PyTorch Mobile, enables developers to deploy any PyTorch model to both Android and iOS. (Figure 12)

Deploying A PyTorch model to Android requires the steps below:

- Convert model to TorchScript format (Python)
- Add PyTorch Mobile as a Gradle dependency (Java)
- Load saved model with PyTorch Mobile to perform predictions (Java).

We want to perform this process successfully and get the apk file. When we consider that our model is generally response quickly, transferring it to an application will be fantastic.

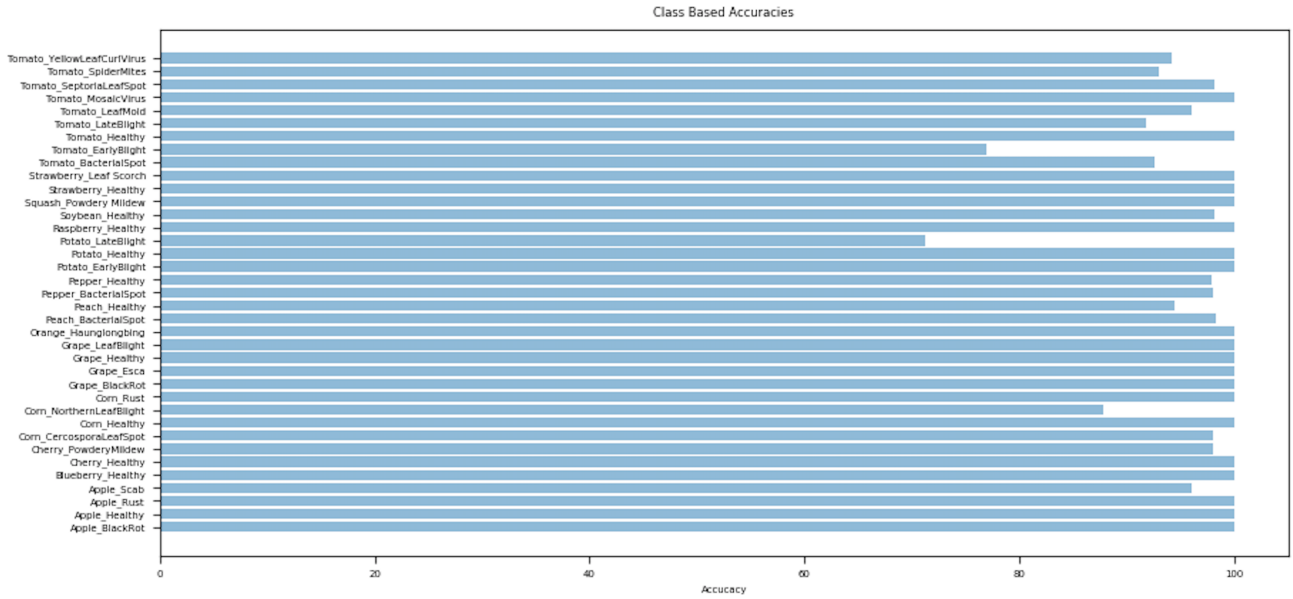


Figure 11: Class Based Accuracies

Figure 12: Mobile application

References

[1] J. K. P. R. Kumar, "Advances in image processing for detection of plant diseases." https://drive.google.com/viewerng/viewer?url=https://pdfs.semanticscholar.org/0a10/96ca51b4b8067865211b42c0bec4c9969f34.pdf, June 2011.

[2] "One of the plant disease datasets." https://www.kaggle.com/saroz014/plant-disease.

[3] M. S. Sharada P. Mohanty, David P. Hughes, "Using Deep Learning for Image-Based Plant Disease Detection," 22 September 2016.

[4] "Adam optimizer." https://pytorch.org/docs/stable/_modules/torch/optim/adam.html.