ENS 491-492 – Graduation Project

Draft Final Report

Project Title: Plant Disease and Identification

Group Members:

Barış Sevilmiş

Cem Alptürk

Kerem Yıldırır

Yiğit Aras Tunalı

Supervisor(s): Ayşe Berrin Yanıkoğlu

Date: 05/04/2019

1. EXECUTIVE SUMMARY

Detection of plant diseases via computer vision based systems are being used to identify plant diseases promptly, to prevent the spread of the disease. In this work, we present a system for classifying plant diseases from photographs of the diseased parts of the plant. The system is trained using transfer learning on convolutional neural networks (VGGNet) which are trained to classify 38 plant diseases or 11 disease classes.

95.09% average accuracy was obtained on plant-disease classification using PlantVillage dataset. On the other hand, 96.27% average accuracy was obtained for disease classification instead of plant-disease classification. VGGNet obtained these scores.

Another main focus of the first part of the project was to implement a web application, using a user interface to make the querying of plant images and getting the results of the disease detection system more user-friendly. We have set up a server in the school network, coupled with a database, which would take queries from users and return the results from our detection system.

Other plans regarding this web application include the gathering of extra data through the user interface and adding the query data, if they are suitable, to the already existing dataset so that the system can be trained further and better results can be achieved in the future. This integration would also make it possible for the system to be improved continuously.

The second part of the project was to integrate the software with an autonomous drone. The objective of the drone would be to scan the field in sufficient time and take close up pictures of the crops. The software would analyze these images and try to classify healthy and sick plants.

With the help of the GPS integrated into the drone, the images classified with plant disease would be saved with the corresponding GPS coordinates and their location would be shown on a map to the user.

2. PROBLEM STATEMENT

The main objective of this project was to be able to develop a machine learning system which could detect, by using an image of the plant, if the said plant had a disease or if the plant was healthy. Detection of a plant using a machine learning system is already a complex task, which makes detecting a disease, notice that each plant may have many different and similar diseases compared to other ones, an even more complicated task. The amount of classification that needs to be done is immense when the aim is to classify a lot of plants and their diseases.

In the literature, there have already been machine learning methods developed that was successful in this task. We adopted the approaches from several research papers and implemented them in our own while adapting the work to fit our needs.

Our primary motivation in this project was to help the automation of agricultural processes. This automation would further develop the methods to grow and produce food more efficiently, which would pave the way to feeding the rapidly increasing population.

2.1. Objectives/Tasks

- **Data Collection & Augmentation:** Since Plant-Disease classification is not a very hot topic as Plant Identification and Face Identification in terms of Computer Vision based Deep Learning methods, finding large image datasets containing plant-disease labels are relatively harder. Therefore, data augmentation stands out as a fundamental objective. Nevertheless, the main objective in terms of data is to find a reliable data source either from external data sources or contribute to the creation of such a database.
- **Model Selection:** While selecting the classification model for our system, we took insights from [3, 5], which were winners of ImageNet Large-Scale Visual Recognition Challenge (ILSVRC), which was a competition on ImageNet dataset. The data set consisted of 22,000 labels, hand labeled by Amazon. The VGG16 model was a winner of this challenge, and we adapted the model structure to our problem and used it for the disease classification since the complexity of our problem was also a very high one.
- **Classification Classes:** As mentioned previously, finding a large amount of data for plant-disease recognition is a difficult task. For this reason, tuning classification classes become a critical choice. Additionally, choosing classification classes is one of the new approaches that we have decided to use over [3, 5], in which we have decided to build half of our models only for disease identification (11 classes), and rest of the models for plant-disease identification(38 classes). Applying only disease detection to some models shines out as an essential objective.
- **Integration with User Interface:** As soon as models are developed and their testing is completed, most accurate model among all of them should be integrated with the User

Interface for the Plant-Disease Application. Learning and using Flask Restful API for a reliable, decent, and successful integration process has the utmost importance.

- **Choosing a vehicle**: To make the crop disease detection work faster, a solution would be to use a vehicle to travel through and scan the crop field. We have decided to use a drone for several reasons:
    - It is much more comfortable to travel through the air than to move on the dirt since the ground level in the field might not be uniform and a ground vehicle may get stuck in the dirt due to an obstacle or merely losing traction due to the mud. Thus a drone will be able to move much faster in the air than a ground vehicle moving through the field.
    - The distance to the crop can be controlled with a drone. Since we need to take pictures of the plants in the field, the leaves or fruits may be located higher for some plants than others. With a ground vehicle, the distance to these cases cannot be controlled where a drone can move in 3 dimensions to get close enough to the plant for a clean image.
    - When taking the images, the vibrations in the environment will affect the quality of the images. Ground vehicles will be induced to vibrations due to the shape of the tires and the shape of the uneven ground. These vibrations and disturbances will require correction for the camera orientation and vibration cancellation. A drone, however, is more stable in the air and can be equipped with a 3 or more DoF (Degree of Freedom) gimble to keep the orientation stable and reduce vibrations.

After considering these reasons, we decided to use a quadcopter/drone for our project implementation. The drone could be constructed and programmed by ourselves or could be bought as a consumer-ready product where no programming is needed / possible. The selected drone is a consumer-ready device, which is a DJI Phantom 3 SE[2].

- **Achieving Autonomous Motion:** Since the main idea of this project is to automate disease detection in crops, the next step would be to use a vehicle autonomously. Our first solution was to use the SLAM (Simultaneous Locating and Mapping)[1] algorithm to achieve autonomous flight. The algorithm will be explained in the Methodology section. After inspecting the complexity and the requirements for this method, we decided to implement this task in a different way using GPS navigation with the help of a 3rd party software. This software will be explained below, as well. By choosing to use a 3rd party software, we can use a consumer drone which is ready to use, and no programming is needed. This will not have been the case if we have used the SLAM[1] algorithm, since we would have to construct and program our drone, consuming more time and resources.

- **Integrating the classifier with the selected vehicle:** A decision had to be made whether the vehicle would perform the classification in real time or after the scanning is complete. Since we decided to use a consumer product to use a 3rd party GPS navigation software as stated above, we were unable to create a real-time system. This way, we implemented our system in a way that, the drone performs a flight above the field while taking pictures /video. After the flight, the video or images are extracted from the drone, together with

the GPS data which can be extracted with a software[3] provided by DJI. The extracted images are overlapped with the GPS data so we can approximately know where the drone was when this picture was taken. After the images are fed through the classifier, the detected diseases are prompted with their given GPS locations and are shown on the map. *last part not completed yet test flight to be performed.

2.2. Realistic Constraints

One of our main constraints is to find plant-disease data to enhance our plant disease identification model. As we already mentioned, there are many different methods to obtain data. However, collecting plant disease images is not an easy job, since capturing a specific plant with a specific disease will probably not occur at the requested time. Geographical conditions play a huge role, as well. Therefore, relying on creating a plant-disease image database would not be a realistic goal, but contributing to its creation would benefit the plant-disease identification process. Exterior data owners are not reliable sources as well, because they may not share their data because of private reasons. Therefore, we are limited to public datasets, Google Images[6]. Another constraint is that Google Images[6] not being a decent plant-disease image data provider, the amount of dirty data is enormous. Unfortunately, there are few public datasets for plant-disease identification.

As a consequence of these constraints, the combination of the above methods(Public datasets, Google Images[6], Exterior Data Owners, and Self-taken images) would benefit at most. Otherwise, the amount of data would be insufficient. Lastly, the usage of drones are among the

best methods to create and contribute to a plant-disease database. However, there are some constraints for drones, following part provides insight about these constraints.

To contribute to plant disease database efficiently, we have decided to use a drone together with a camera. For this process to work, the camera must capture the plants in high quality and without motion blur. This requires a high-quality camera. Another problem with the camera is the weight. The extra weight of the camera can cause the drone to draw more current from its battery and decrease the flight time or even worse, can result in the drone being unable to fly. Apart from the physical issues, the Turkish Government has restrictions on drones. According to the Civil Aviation laws [4], if the weight of the drone is over 500 grams, it must be registered, and a permit is needed before every flight. This will make the inspection much more difficult. So According to these constraints, the camera must be compact, low weight, and be able to capture high-resolution images.

Computational power to build up deep learning systems stands out as another realistic constraint. Although deep learning systems do not require high-performance machines, it is crucial to use high-performance clusters to train models more efficiently and faster. Unfortunately, our local computers do not provide high processing powers in terms of Graphical Processing Units(GPUs). Therefore, finding alternative high-performance sources becomes a necessity. Google provides its computational power sources, Tesla K80 GPUs, to the public through Google Colab freely. However, these GPUs cannot be utilized by 100%. For this reason, model training does not achieve maximum efficiency. As a consequence, Google Colab should be used if high-performance alternatives cannot be afforded, or High-Performance Clusters should be

taken into consideration. We were provided with High-Performance Clusters, though our first set of model training was done on Google Colab as HPC were not initially ensured.

3. METHODOLOGY

3.1 DATA COLLECTION & AUGMENTATION

Figure 1: Plant Village Dataset

In the previous part, we already mentioned the realistic constraints of the project, in which gathering a sufficient amount of useful data is not an easy task. Fortunately, Plant Village dataset was found as an exterior public dataset.

PlantVillage data set consists of 54.305 images from 14 different plant species. There is a total of 38 disease-plant pairs(for example apple rust) in the data set. A slice of the data set is shown in the figure below. We split 20% of this data set for testing and did our training using the VGG16 network. Dealing with an insufficient amount of data was the most problematic part of this project.

Figure 1 demonstrates a distinct part of the Plant Village dataset, where each column represents a different plant, namely from left to right: apple, cherry, grape, corn, pepper, tomato, strawberry, peach. However, this only a tiny part of Plant Village Dataset.

To build a successful deep learning system, it is required to train the system with a large amount of data. Although 54.305 images seem like a large amount of data, it is not sufficient to build a realistic model that can identify plant-disease labels. Nevertheless, establishing an accurate deep learning system with the data at hand would still yield a successful model. In future works, with more data, our core deep learning systems could be chosen as the first checkpoint and further improved. However, again for a valid core deep learning system, a sufficient amount of training data is required.

As a consequence, we have decided to use data augmentation, in which on fly augmentation was used. In other words, augmentation was applied during the training process. Besides, basic augmentation methods such as shifting, rotation, and translation were applied to Plant Village images using Tensorflow and Keras.

3.2 CONVOLUTIONAL NEURAL NETWORKS

3.2.1 VGG-16

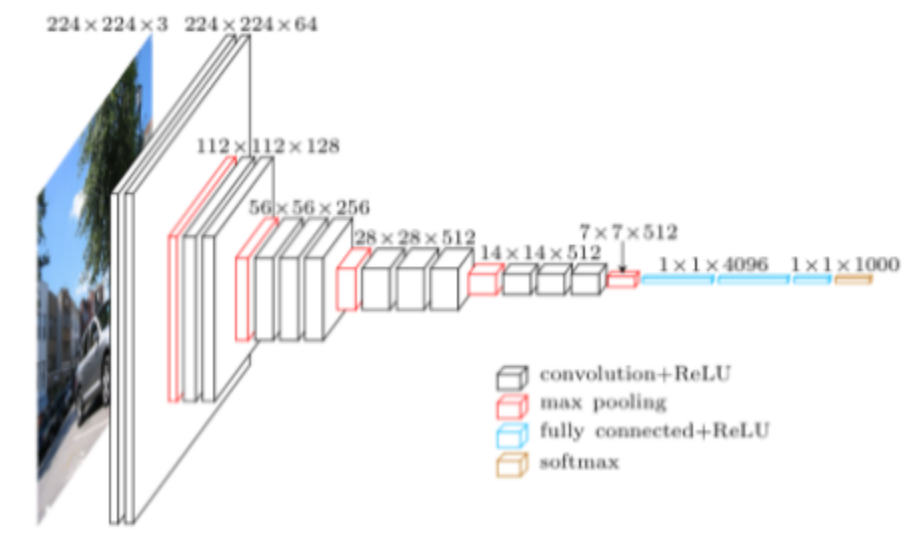

Figure 2: VGG-16

In this work, we used deep learning, a method which became popular in recent years due to the increase in the amount of data and computation power. Deep learning is primarily used in computer vision problems such as classification and object recognition. There are also annual challenges for deep learning applications where people from different universities compete for the best classification model. One of those competitions, ImageNet, is a challenge for object recognition, and it is one of the most famous challenges. We thought that the problem in the challenge is similar to our problem and decided to finetune VGG-16, one of the deep learning architectures which performed well in the ImageNet challenge. To finetune the VGG16 architecture(can be seen below), we replaced the last fully connected layer with another fully

connected layer with 38 units and softmax activation in case 1 and a fully connected layer with 11 units with softmax activation in case 2. We trained these last layers from scratch, while we froze the other layers so that the number of trainable parameters kept at a minimum as we do not have much computing power. We used 20% of our training data for validation and used it as the early stopping technique through 100 epochs. Figure 2 depicts the general structure of the VGG-16 model, and Figure 3 demonstrates the logical structure of VGG-16 model.

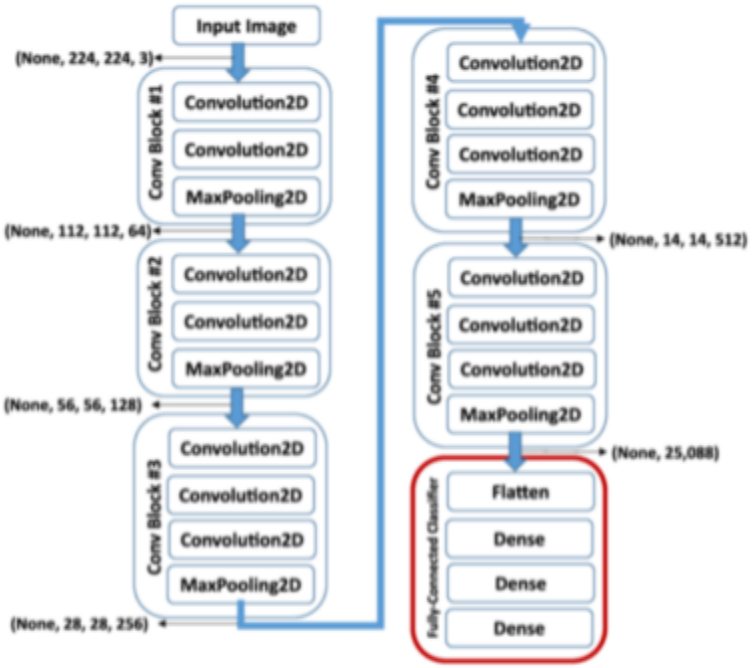

Figure 3: Logical Structure of VGG-16 model

3.3 DRONE INTEGRATION

3.3.1 SLAM

SLAM is an algorithm that allows a mobile robot/vehicle to create a map of its environment and calculate its position and orientation according to this map simultaneously. It does this with the combination of many onboard sensors such as accelerometers, sonar sensors, lidar sensors, gyroscopes, and most importantly, stereo cameras. This is a very complicated process since it only uses its sensors that are on the drone, and all the data that is processed includes noise. As stated above in the objectives section, such an algorithm has to be implemented on a self-made drone, and this was out of the scope of our project. Due to these reasons, as explained above, a consumer-ready drone with GPS position control was selected.

3.3.2 DJI Phantom 3 SE

As stated above in the objectives section, we have decided to use *DJI Phantom 3 SE*. The technical specs of the drone:

- Take off weight of 1.1236 kg
- Max horizontal speed of 16 m/s
- Hover accuracy range of ±0.5 m (vertical), ±1.5 m (horizontal) GPS positioning
- Max hovering time of 25 minutes
- Max service ceiling of 6000 m above sea level
- 3 axis camera stabilization

- Electronic shutter speed of 8-1/8000 seconds
- Image size of 4000x3000 pixels
- FOV 94 degree 20mm f/2.8 focus at ∞ lens
- Live video quality of 720p with 30 fps

3.3.3 GPS Availability

The drone records it's flight data on to its internal storage and can be received by using the DJI GO app and entering into Flight Data Mode [5]. By connecting the drone to a PC, the log files of all previous flights will be available. After extracting the flight logs, the log viewer website[3] is used to extract the GPS coordinates in CSV format.

3.3.4 Fixed Height and Route

The third party software Litchi[6] is a third party autonomous flight app and is compatible with DJI Phantom 3 SE. The app has a mission planner that allows the user to enter a path, orientation, speed, camera angle, and altitude that the drone will follow during the mission. The waypoints for the mission can be set from the app or the website[6] since it uses images from Google Earth, we can visually see the landmarks that will make the planning easier. For our problem, we created waypoints across the crop field and gave the drone a fixed altitude and a fixed gimbal orientation with the desired speed. During the flight we can observe the live camera feedback from a tablet or smartphone in case something went wrong.

3.3.5 Approximating Area Covered per Flight and Timing

As stated above, the DJI Phantom 3 SE has a FOV of 94 degrees. If the drone camera is oriented normal to the ground and the drone is flying at altitude h meters relative to the ground, the size covered by a frame of the video will be approximately 2h meters wide.

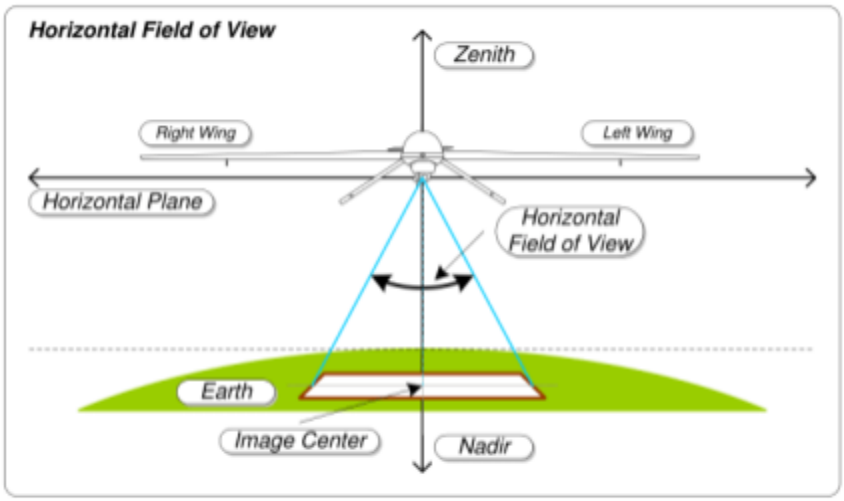

Figure 4: Field of View(FOV) illustration

We can approximate the area covered by the frame as a circle with radius h or a square with each side being 2h meters. If we go with the second approximation, a single frame will cover an area of 4h2m2. Assuming a single plant in the field covers around 1m2 and is positioned in the center of the image, the drone would need to fly at least 1m above the ground, which is too low. If the drone flies 2m above the ground, the plant will cover approximately 1/16'th of the frame, and the drone will have less probability to collide with objects. Since the drone uses GPS for vertical positioning and has a ±0.5m error rate, the optimal height for the drone will approximately be 2.5-3m. The height has been tuned during the test flight.

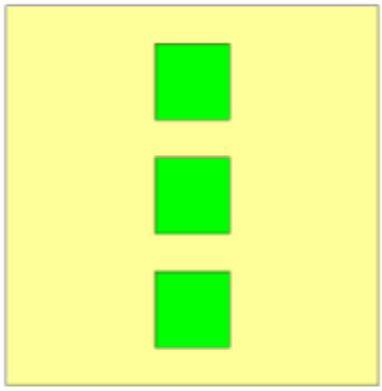

Figure 5: Frame area(25m2) vs plant area(1m2) when h=2.5m.

Assuming each frame will cover 4h2m2 of area, with non-overlapping frames, the drone can scan the field in a column-row wise manner. Assuming a square field with Lm sides and each pass over the field will cover L/5 * 25=5Lm2 where each non-overlapping frame covers ~25m2 flying at 2.5m. In total, the mission will require L/5 pass-by's each covering 5Lm2 of area.

For a field with 104m2 area, the drone will have to do 20 passes if we assume the frames are non-overlapping. With overlapping frames, the number of passes will increase, and the same plant will be captured multiple time, which can increase the accuracy of the disease being detected since a different angle is introduced. However, since the drone will be recording a video, ~30fps (can be set up to 60fps)will be the standard. The video can be processed into frames after the flight.

A field of 50x50m2 with each plant covering 1m2.

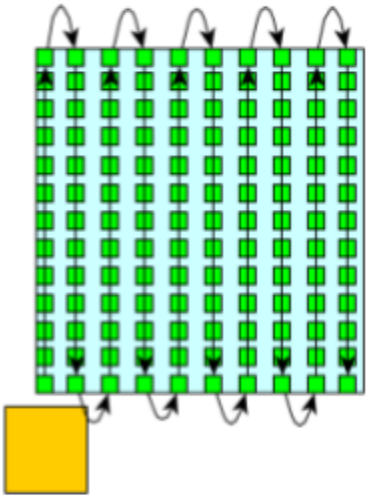

Figure 6: A field of 50x50m2 with each plant covering 1m2. The yellow block representing the field covered by the camera in a single frame.(h=2.5m)

The flight time of the drone is limited to 25 minutes, so the velocity of the drone must be adjusted. We need to be fast enough for the battery, but we also need to be slow to obtain clear footage. If the drone is flying with a constant 2m/s velocity, it will take 50s for a single pass and 4 passes to entirely cover the entire field, assuming the field is 100x100m2. Thus with a relatively slow velocity, we can cover a hectare in ~5 minutes. Which means the battery will last for most of our missions. The velocity of the drone can be lowered to get more clear images if necessary. It can be lowered as low as 0.3m/s to cover a hectare field in under 25 minutes.

3.4 USER INTERFACE WITH REST API

One of the primary purposes of the project was to ensure a useful interface for Plant Disease Detection system. As there already existed Plant Identification User Interface belonging to Sabancı University, the best idea was to extend this User Interface with Plant Disease Identification optionality. Accessibility to both Plant Identification feature and Plant Disease Identification feature would, therefore, be more practical and reasonable.

Flask-Restful API was used to integrate the Plant Disease model with an already existing interface. When an image is entered, the model predicts plant-disease label of the given image by accessing image through SQL database. After an image is uploaded to the website, it is directly sent to the SQL database. With the help of SQL queries, model accesses given image and predicts the corresponding label. Prediction is printed out on the website.

3.5 USAGE OF GOOGLE COLAB

Google Colab was used as a working environment to create Plant Disease models. By free GPU usage provided by Google Colab, models were trained relatively faster in comparison to training with CPU's. Also, Jupyter environment provided by Colab assisted us to build our models separately without losing the modularity of the code as different model training only required working with different code blocks. Executing non-relative blocks were avoided. Lastly, steps, as downloading data or importing libraries, were done only once, since they were distinct code blocks. Therefore training and testing models gained momentum in comparison to using Python with non Jupyter environment.

4. RESULTS & DISCUSSION

As mentioned in previous sections, we did build four VGG-16 models with different characteristics, to be more specific, we trained our first two models to detect only diseases but not both plant-disease labels. Therefore, these two models were trained only for 11 classes, and 11-node softmax layer was used to enable classification. Other two models were built for plant-disease classification, namely 38-node softmax layers for classification. Lastly, the difference between models having the same classification tasks was their trainable layer amount. Two of these models, one from 11-class and other from 38-class, were changed, such that their fully connected layers were extracted and three-layer from their remaining layers were enabled for training. In other words, transfer learning was applied, and layers. Other two models were the same, only that their last single layers were enabled for training. All of these models were trained for 100 epochs, and most accurate model was the 11-class model with 3-trainable layers. Our accuracy obtained by this 11-class model was 96.27%, and our best accuracy obtained by 38-class models was 95.09%.

Considering two models that were built for 11 classes, highest and lowest F1-scores achieved were 99.53% and 89.95% (96.64% on average) on the basis of classes. Highest and lowest accuracy values resulted as 100% and 88.50%. Lowest accuracy value (88.50%) belonged to mold disease, however mold image dataset consists of very few images and for this reason, we

believe that F1 and accuracy scores for mold can be increased through a greater image database.On the other hand, highest and lowest F1-scores achieved by 38-class models were 99.77% and 81.40% (94.79% on average).

Loss and sparse categorical accuracy graphs of the most accurate model, 11 class model with the last three layers trainable, are depicted below in Figure 7 and Figure 8. Until 20 epochs, a decrease in loss value and an increase in accuracy indicates around 20 epochs are required for decent training. To avoid overfitting, we have used our validation set such that after every epoch validation set was tested. Only if validation accuracy was higher than the previously saved accuracy value, then these new epoch results were stored. For the rest of the epochs, their changes in later weights were not stored.

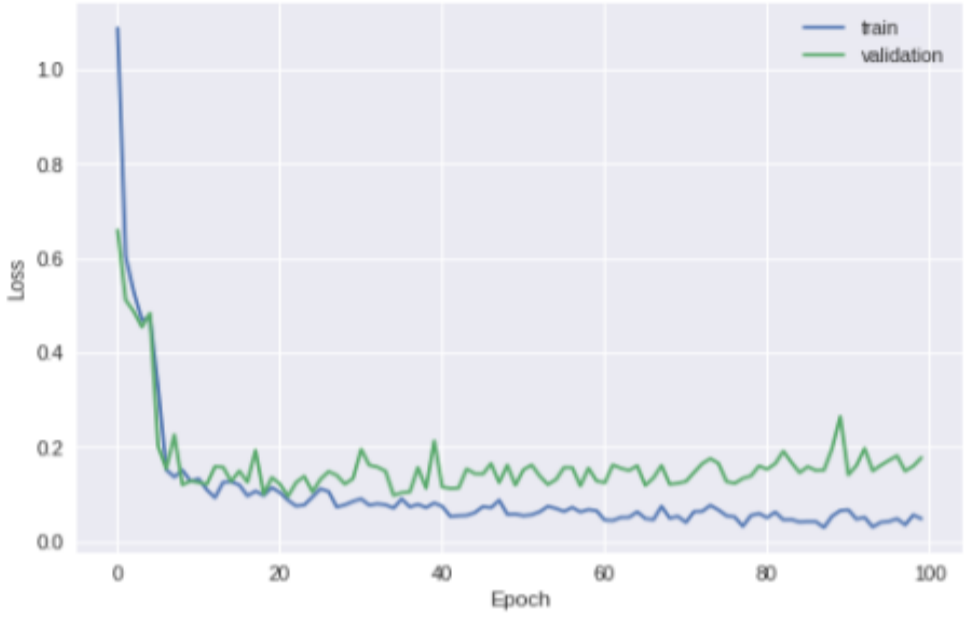

Figure 7: Change of Loss value in 100 epochs

Figure 8: Change of Accuracy in 100 epochs

We have achieved our initial expectations, as our models have achieved decent F1 and Accuracy scores. We used the state of the art technology to develop these models such as Tensorflow and Keras. Besides, we believe that our models contributed to theses state-of-art technologies and agricultural laborers themselves. Even our contributions are not yet a factor of influence. We believe that in the near future they have a high chance to be guidance to similar works.

The drone integration part is still ongoing, and a test flight will be performed soon. Since the test flight will be performed on a garden rather than a crop field, we do not expect correct classification because the plants located in the garden are not included in the training set. Thus the test flight will be focusing on the autonomous flight, mission planning, optimal flight velocity, optimal hovering height, optimal camera gimble orientation, data extraction, and processing.

Table 1: 11-Class Result Table(Precision, Recall, Accuracy, F1-Score)

Hastalık Adı	**Kesinlik**	**Anma**	**Doğruluk**	**F1**
blight	91.99	94.54	91.99	93.24
greening	99.81	99.91	99.81	99.86
healthy	99.13	97.15	99.13	98.13
measles	93.68	99.60	93.68	96.55
mildew	99.66	98.47	99.66	99.06
mold	88.50	91.24	88.50	89.85
rot	95.05	97.19	95.05	96.11
rust	98.97	93.51	98.97	96.16
scorch	100.00	99.07	100.00	99.53
spot	93.27	94.94	93.27	94.10
virus	98.95	97.50	98.95	98.22
Ortalama	**96.27**	**96.65**	**96.27**	**96.44**

Table 2: 38-class Result Table (Precision, Recall, Accuracy, F1-score)

Name of the plant disease	Precision	Recall	Accuracy	**F1**
Apple___Apple_scab	95.12	90.70	95.12	92.86
Apple___Black_rot	97.56	94.49	97.56	96.00
Apple___Cedar_apple_rust	81.13	97.73	81.13	88.66
Apple___healthy	98.19	97.31	98.19	97.74
Blueberry___healthy	99.09	98.20	99.09	98.65
Cherry_(including_sour)___Powdery_mildew	97.71	99.53	97.71	98.61
Cherry_(including_sour)___healthy	97.34	98.39	97.34	97.86
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot	87.96	83.33	87.96	85.59
Corn_(maize)___Common_rust_	99.16	100.00	99.16	99.58
Corn_(maize)___Northern_Leaf_Blight	89.95	91.89	89.95	90.91
Corn_(maize)___healthy	99.52	99.52	99.52	99.52
Grape___Black_rot	98.76	98.76	98.76	98.76
Grape___Esca_(Black_Measles)	98.51	98.88	98.51	98.70
Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	98.19	99.09	98.19	98.64
Grape___healthy	98.80	98.80	98.80	98.80
Orange___Haunglongbing_(Citrus_greening)	100.00	99.54	100.00	99.77
Peach___Bacterial_spot	98.67	98.89	98.67	98.78
Peach___healthy	95.65	97.06	95.65	96.35
Pepper,_bell___Bacterial_spot	96.02	93.24	96.02	94.61
Pepper,_bell___healthy	98.38	97.43	98.38	97.90
Potato___Early_blight	88.46	99.46	88.46	93.64
Potato___Late_blight	89.06	89.53	89.06	89.30
Potato___healthy	94.59	71.43	94.59	81.40
Raspberry___healthy	95.74	98.90	95.74	97.30
Soybean___healthy	99.42	99.61	99.42	99.52
Squash___Powdery_mildew	98.90	100.00	98.90	99.45
Strawberry___Leaf_scorch	99.53	99.07	99.53	99.30
Strawberry___healthy	100.00	96.51	100.00	98.22
Tomato___Bacterial_spot	96.49	92.11	96.49	94.25
Tomato___Early_blight	76.17	88.11	76.17	81.70
Tomato___Late_blight	94.56	84.18	94.56	89.07
Tomato___Leaf_Mold	90.50	90.95	90.50	90.73
Tomato___Septoria_leaf_spot	93.01	91.53	93.01	92.27
Tomato___Spider_mites Two-spotted_spider_mite	95.28	93.08	95.28	94.17
Tomato___Target_Spot	86.17	91.35	86.17	88.69
Tomato___Tomato_Yellow_Leaf_Curl_Virus	97.58	99.71	97.58	98.64
Tomato___Tomato_mosaic_virus	96.97	82.05	96.97	88.89
Tomato___healthy	95.37	99.04	95.37	97.17
Average	**95.09**	**94.72**	**95.09**	**94.79**

5. IMPACT

We believe that our research product can be implemented in a mobile application which could be used by farmers and agricultural engineers for disease identification. Given sufficient data, we believe that our research product can be implemented in a mobile application which could be used by farmers and agricultural engineers for disease identification. Given sufficient data, we believe that our research product can be implemented in a mobile application which could be used by farmers and agricultural engineers for disease identification. Given sufficient data, we believe that our research product can be implemented in a mobile application which could be used by farmers and agricultural engineers for disease identification. Given sufficient data, it is possible to train better models with more accuracy in the field. The current version is constrained by the small amount of lab-only data. Hence for it to be used in the field, we need more data.

6. ETHICAL ISSUES

There are no ethical issues regarding this project.

7. PROJECT MANAGEMENT

We started by collecting data from google images as we were unable to gather relevant data immediately for our project. However, the data we have gathered was not nearly enough to train our models. Hence we decided to use a dataset from a challenge in this domain. We continued our project with the implementation of a paper which used our dataset and gathered good results. After successfully training the model and getting similar results to the paper[5] we moved to the drone integration. We are currently in the phase of collecting data with the drone, and we plan to use this data further training and improvement.

8. CONCLUSION AND FUTURE WORK

As a result of this project, we have developed a machine learning application for potential use in agricultural automation. We plan to develop it further such that it is easily accessible and beneficial to farmers and agricultural engineers. Our work on the drone is still ongoing, and we aim to complete the integration as the final step. Future work that could be done on top of this project is evident. If given more and accurate data, there is much room for improvement in the model training.

9. REFERENCES

[1] K. Berns and E. V. Puttkamer, "Simultaneous localization and mapping (slam)," Autonomous Land Vehicles, p. 146172, 2009.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," CoRR, vol. abs/1409.1556, 2014.

[3] Mohanty, Sharada, D. P., Marcel, and Hughes, "Using deep learning for image-based plant disease detection," Sep 2016.

[4] Wang, Guan, Sun, Yu, and Jianxin, "Automatic image-based plant disease severity estimation using deep learning," Jul 2017.

[5] K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," Computers and Electronics in Agriculture, vol. 145, pp. 311 – 318, 2018.

[6] K. G. Dangtongdee, "Plant identification using tensorflow."

[7] https://www.dji.com/phantom-3-se

[8] https://www.phantomhelp.com/LogViewer/Upload/

[9] "Turkey drone laws," July 2018.

[10]https://support.dronesmadeeasy.com/hc/en-us/articles/206171443-Flight-Data-Log-Viewer-DJI-Phantom-3-and-Inspire-1

[11] https://flylitchi.com/help