A SMART PLANT MOISTURE LEVEL DETERMINATION SYSTEM TO DETERMINE IF THE PLANT NEEDS TO BE WATERED OR NOT BY USING MACHINE LEARNING

Ruohan Zhang[1], Yaotian Zhang[1], Yu Sun[2]

[1]Fairmont Preparatory Academy, 2200 W Sequoia Ave, Anaheim, CA 92801
[2]California State Polytechnic University, Pomona, CA, 91768, Irvine, CA 92620

ABSTRACT

All living things including plants need water to survive, and agriculture is the world's biggest user of water [4]. Unfortunately, in a worst-case scenario, over-watering and drying up cause both water waste and the plant's death [5]. Guided by this problem that is frequently occurring around the world, we designed an app to determine if the plant needs to be watered or not by capturing pictures of a certain plant and training an AI to compare whether the soil in the pot is dry or wet. In this program, we use Raspberry Pi to capture an image of the plant every 10 seconds,in which the Python code using TensorFlow inside the Raspberry Pi will determine the moisture level of the soil [6][7]. The result will be posted to Firebase with a timestamp, and lastly, we have a mobile app that can display the result from Firebase to the user. We published our application to Apple's App Store and the Google Play Store, and public installation of the app means that it can have more widespread usage. An experiment was performed to determine whether the application's model can accurately determine soil moisture [8]. The results indicate that the model is very accurate for the vast majority of soil samples under various lighting conditions.

KEYWORDS

Python, Flutter, Machine Learning, Firebase

1. INTRODUCTION

If you ever noticed your plant is turning yellow, it is possible that your plant is being overwatered. Overwatering is one of the most common causes of plant problems [9]. Overwatering severely limits the supply of oxygen that roots depend on to function properly, meaning that plants do not get enough oxygen to survive. Also, if the soil is heavily drained, it will become waterlogged and the roots growing in this soil will die [10]. Furthermore, overwatering can lead to broader problems such as the over usage of water. Take California farmers as an example. In the year 2021, California's farmers pumped an additional six to seven million acre-feet of water from their wells above what they normally use. This quantity of water would cover 10,000 square miles with a foot of water. Problems related to overwatering are happening in a lot of parts of the world, from one's backyard garden to a local farm. So, in what ways can people prevent a plant from being overwatered or dried out?

Some people come up with a sensor that can detect the moisture level of soil for the purpose of avoiding overwatering. However, these sensors assume that the owner only has a small number

David C. Wyld et al. (Eds): CCNET, AIMLA, CICS, IOTBS, NLTM, COIT - 2023
pp. 127-134, 2023. CS & IT - CSCP 2023 DOI: 10.5121/csit.2023.130410

of plants, which is not the case. Normally, people with large farms tend to need these sensors more because they can't take care of every plant that well and make sure they are healthy. To make sure every part of the soil on the farm is healthy, they will need to buy hundreds and thousands of them, which leads to the second issue. A second practical problem is that the sensor needs to be taken care of. These sensors detect the moisture level by directly inserting them into the soil and waiting for a few minutes then pulling them out. Let's use the case as if you own a large plantation and are rich enough to afford to buy many of these sensors. You will need to insert them one by one and sit there waiting for the sensor to work. Then, you will need to record the mixture level for each area you measure and determine whether a certain area needs to be watered or not. Next, you will go around your plantation again to pull each sensor out of the soil, and lastly, you need to clean them for them to work again. This process takes both time and effort, which are both valuable. Seeing these issues with existing tools, we came up with our topic of creating an app that can monitor your plant and notify you when the plant needs to be watered, which can avoid both overwatering and drying out of the plant.

The purpose of our application is to predict and provide a real-time moisture level of plants and avoid overwatering or withering. To provide an accurate estimate, the application uses many steps to make predictions. Firstly, the application gathers the plant's picture by using a small Raspberry Pi camera that the user can operate simply. Secondly, the Raspberry Pi camera sends the picture to Raspberry Pi, where our program analyzes and processes the image. Thirdly, Raspberry Pi sends the results and analyzed data to Firebase, where our server is built and data is stored in [11]. Lastly, Firebase returns the results to the user's mobile application where the user can access the data. However, only following a specific order to process the image in some cases won't always be accurate, instead, we also used a machine learning process to improve the application even more. Compared to other moisture sensors, our sensor requires much fewer conditions to run accurately, for example, our sensor can take pictures of large amounts of soil by simply pointing the camera to it. The sensor analyzes the soil as soon as the user takes the picture. While some other moisture detectors requires other more complicated steps like pointing a long iron stick, our method has a much simpler and more streamlined process.

The effectiveness of the application can be measured by the accuracy of the application in determining the soil moisture of a given sample of soil. Implementing a smaller-scale experiment to start with can pave the way to future larger-scale experiments after the necessary adjustments to the application have been made. The experiment involves 20 different soil samples, of which 10 of them are dry and 10 of them are thoroughly watered. Using the application, each sample will be analyzed for its soil moisture by taking a picture of it from a top-down angle. Having all the pictures taken from the same angle can reduce confounding variables in the experiment. The total number of dry soil samples and the number of wet soil samples that were correctly identified will be recorded in a table. The goal of this experiment is to ensure that the basic code and model within the application work as intended so that more adjustments and expansions can safely be made to the application in the future. If almost all of the soil samples are identified correctly by the application, the code and model will need no further adjustments in the near future, and effort spent on improving the application can be applied to other aspects of the application instead. However, if a significant portion of the samples is incorrectly identified by the application, then the most urgent change to make would be creating a new model or adjusting the current model. By progressing with the application, it can hopefully find practical use in the field of agriculture.

The rest of the paper is organized as follows: Section 2 gives the detail on the challenges that we met during designing and developing the application and the experiment to test the effectiveness of the application; Section 3 focuses on the detail of our solution related to the challenges that are mentioned in section 2; Section 4 presents the details about the experiments we did and the

related works will be presented in Section 5; lastly, Section 6 provides concluding thoughts regarding the project as well as a brief self-reflection to see what could be improved on in the near future.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Deciding what angle to approach the problem with

The first obstacle with the project was deciding what angle to approach the problem with. Providing plants with an improper amount of water to cause overwatering and drying is a relevant issue, so a reasonable solution would be offering people a method to measure the moisture of a given area of soil. To do this, a sensor could be used to take a picture of the soil. Using the picture of the soil, a system would need to determine the soil moisture from the picture and return the result to the user in a convenient manner. The simplest solution to do so is a mobile application. Most people carry a phone around nowadays, which means that almost everyone will be able to easily access the application. The main overall concept behind the mobile application is retrieving the soil moisture from the sensor and the back-end code, then printing the predicted soil moisture to the screen, which can inform users of the application and help them determine whether or not they should keep watering a plant.

2.2. Creating the code that would be used in the application

The next challenge is creating the code that would be used in the application. The purpose of the code would be to retrieve a picture of the soil and make a prediction as to whether the plant has been properly watered or not. To do so, a sensor that acts as a camera will constantly check what it is currently seeing for soil moisture by using a while True loop. By using a while True loop, the code ensures that as long as the application is running, the application will constantly run the lines of code to update the image every 10 seconds with the most recent image that the sensor detects, then loads a new model to run through the updated image with. Another issue with the code is that the front-end is made with Thunkable and the back-end is coded using Flutter. To combine the two different programming languages, a Flask server is used; the Flask server helps HTTP requests move back and forth. Furthermore, Firebase also helps with the transferring of image files.

2.3. Figuring out how to experiment with the system

The final obstacle was figuring out how to experiment with the system. The ideal experiment would be testing with multiple sensors across a wide area of land for soil moisture. However, the application is currently only capable of supporting one sensor at a time, and access to much farmland is costly and difficult to acquire. Furthermore, before making such a large-scale experiment, determining whether the application is reliable at accurately gauging soil moisture on a smaller scale is an important step. Therefore, the experiment that was decided upon was to take ten samples of dry soil and ten samples of moisturized soil, use the sensor to observe the predicted soil moisture levels from the application for each sample, and record the results on a table. If the application was able to accurately identify the soil moisture for the vast majority of the soil samples, it could be concluded that the application would make for a reliable product.

3. SOLUTION

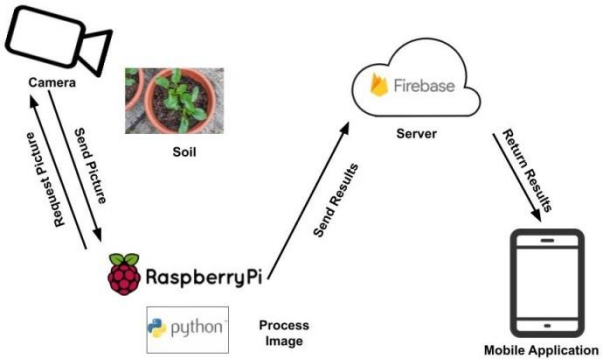

Figure 1. Overview of the solution

The system of the application involves a camera, a Raspberry Pi, a Firebase server, and the mobile application itself. A Raspberry Pi is a small and low-cost single-board computer that is capable of requesting and retrieving pictures from a connected camera. The camera is placed to face a soil sample, and the camera takes a picture of the soil and sends it to the Raspberry Pi whenever it is instructed to do so. The Raspberry Pi runs Python code that acts as the back-end of the mobile application and runs a model to determine whether the soil sample that is captured in the picture is dry soil or wet soil, using the image retrieved from the camera as input for the model [12]. As Python is a popular and relatively simple language in terms of syntax, there are multiple packages to choose from dedicated to artificial intelligence and image classification. After the model finishes processing the image, the results are sent to the Firebase server. Finally, the results are retrieved from the server and printed in the mobile application. The front-end of the application is created with Thunkable, which is a platform that prides itself on its simplicity and allows the building of mobile applications with little to no code; rather, Thunkable users can drag and drop any components of the user interface that they desire. Within the application, the information that is projected to the screen is the classification of either wet soil or dry soil, the percentage of confidence from the model, and the image that was taken from the camera.

Python was selected to be the code for the back-end; it is not only easy to work with due to its relatively simple syntax but also convenient to achieve specific features due to the myriad of packages that are available. To create the back-end of the application, the Python code was separated into four separate Python files. The first Python file is for the application itself. Within this file, the cv2 file is used to perform video capture [14]. Then, a while True loop is made to indicate that the code within this loop should continue to run as long as the application is active and running. Variables collect what is currently being read from the camera, and the timestamp is collected as well by using Python's time package. Every 10 seconds, the old timestamp will be replaced with the new, current timestamp, and a new image will be retrieved. Using the path of the new image, the predict method from the classifier file is called. The predict method makes use of three other methods; one of them loads a model, the second one loads the labels, and the final one loads the image. Using the package TensorFlow, a model is loaded in and allocated tensors [13]. Then, the input and output tensors are retrieved and returned within the method to load a tflite model. For the method that loads labels, the path to the labels is opened and read, then compiled into a list and returned. The final method to load the image uses an image path and specifies that the target size of the image should be 224 pixels by 224 pixels. The loaded image is then converted to an array, has a batch created for it, and then is returned. The predict method loads the model, loads the label, and loads the image in that order. After the input tensor is set, the inference is run by calling the invoke method on the model, then the prediction is

retrieved by retrieving the tensor and taking the one with the highest confidence.

Thunkable is used as the front-end, and the application is divided into two separate screens. The first screen simply has a logo object that acts as an introductory splash screen. The second screen is the main screen that reveals all of the needed information to the user. There are four components on this screen, which are the soil label, the time last updated in seconds, the confidence label, and the captured image by the camera. The soil label states whether the soil sample was determined to be a “Wet Soil Pot” or a “Dry Soil Pot”. The confidence label states how confident the model was in its prediction as a percentage. The Firebase real-time database is indicated within Thunkable as an invisible component, which is how the components on the second screen are able to retrieve the necessary information.

```python
@app.route('/')
def index():
    #return render_template('index.html', filename='uploads/endgame_trailer_final.mp4')
    return 'Soil Image Classification'

@app.route('/classify_image', methods=['GET', 'POST'])
def classify_image():
    if request.method == 'POST':
        f = request.files['image']
        print(f)
        filename = 'images/'+ f.filename
        print(filename)
        f.save(filename)
        pred = classifier.predict(filename)
        if pred != None:
            os.remove(filename)
            return json.dumps(pred)

        return 'Image cannot be analized'
```

```python
def upload_file(self, storage_filename, local_path):
    bucket = storage.bucket()
    blob = bucket.blob(storage_filename)

    if blob.exists():
        print('This file already exists on cloud.')
        return blob.public_url
    else:
        outfile = local_path
        blob.upload_from_filename(outfile)
        with open(outfile, 'rb') as fp:
            blob.upload_from_file(fp)
        print('This file is uploaded to cloud.')
        blob.make_public()
        return blob.public_url
```

Figure 2. Pictures of the Python back-end code

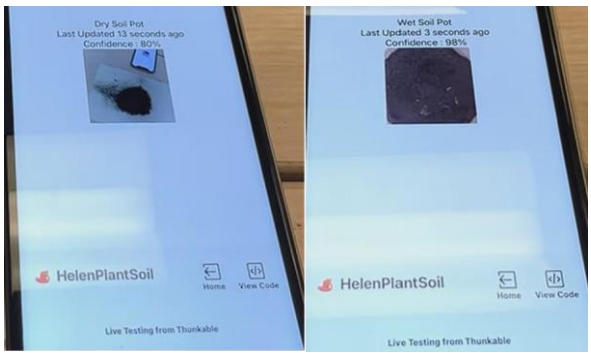

Figure 3. Pictures of the mobile application

4. EXPERIMENT

An experiment was performed specifically to test the accuracy of the model in correctly identifying the moisture of soil in a given soil sample. Ten wet soil samples and ten dry soil samples were gathered, making twenty different soil samples in total. To reduce confounding variables regarding the samples themselves, each sample used approximately the sample amount of soil, and each wet soil sample was watered with approximately the same amount of water. Then, the sensor in the application system was used to take pictures of the soil from a top-down angle, in which taking the pictures of all the samples from the same angle further reduces confounding variables by keeping each sample as consistent as possible. After all the samples have been tested for their soil moisture, the number of correctly identified samples is recorded in a table.

Soil Moisture Type	Number of Correctly Identified Samples	Number of Total Samples	Accuracy
Wet	10	10	100%
Dry	9	10	90%

Figure 4. Table of experiment 1

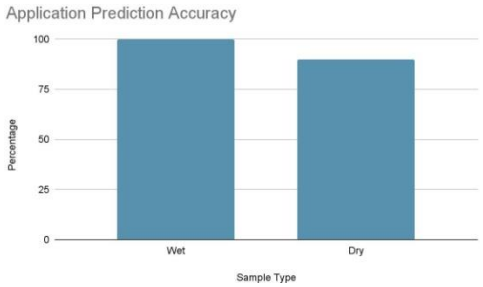

Figure 5. Application prediction accuracy

The application scored highly on its accuracy of both dry soil samples and wet soil samples. The experiment indicated that the application has a 100% success rate when it came to identifying wet soil samples. However, the application only scored a 90% success rate when testing dry soil samples, in which one of the soil samples was identified incorrectly as a wet soil sample. A possible explanation as to why one of the dry samples was identified incorrectly was because of inconsistent lighting throughout the duration of the experiment. The samples were tested outdoors, and the sun changes position and casts different amounts of light on the soil samples throughout the day. From the samples that were tested, the wet samples were noticeably darker than the dry samples. Because consistent lighting was not managed in the experiment, inaccuracies may be caused by the model not being well-trained enough at identifying samples under different levels of lighting.

The experiment was designed to test the model within the application for its accuracy in identifying the moisture in samples of soil, and the results of the experiment would indicate whether the model needs improvement and whether more efforts should be focused on improving the model or improving other aspects of the application. According to the results of the experiment, the model appears to do well at determining the moisture of both wet and dry soil samples. This falls within expectations, as the model has already undergone much training with various soil samples and was expected to perform fairly well. As previously mentioned, something that was intentionally left as a possible confounding variable is the lighting in the

pictures of each sample. In a real-life agricultural setting, the lighting of the soil when detected by sensors would not always stay the same, and the application would need to be tested for its ability to handle different levels of lighting as a result.

5. RELATED WORK

A related work describes different methods that are used to evaluate the moisture of soil, which vary depending on the application or setting that it is used. Therefore, a concept was proposed so that different approaches could be combined into a single integrated system that could be potentially used as a multipurpose solution [1]. This work is similar to the related work in that the primary focal point is analyzing the moisture of the soil. However, the related work goes into depth on various methods that could be used to quantify the moisture of the soil. On the other hand, this work emphasizes the creation of an application to easily gauge whether the soil is properly watered or not.

In another related work, different types of sensors are compared regarding their ability to accurately measure soil moisture. Between the TDR-based sensor that made use of the travel time of an electromagnetic pulse to propagate along sensor rods and the 10HS sensor that worked through capacitance, it was concluded that both types of sensors had their shortcomings and neither one definitively outperformed the other [2]. The related work and this work share the similarity of determining soil moisture. However, while the related work compares how effective different sensors are at evaluating soil moisture, this work focuses on incorporating a sensor into an application.

A third related work experiments on the application of using soil moisture in a drip irrigation automation system that was primarily composed of a base station unit, a valve unit, and a sensor unit. The system was tested on an 8-decare area with dwarf cherry trees, and it was observed that the system was low-cost and reliable and could have practical agricultural use [3]. Both this work and the related work were similar in that the goal was to create an application using sensors to detect soil moisture that would hopefully have practical use in agriculture. However, the related work involves performing a large scale experiment on a wide area of land while this work aims to test the accuracy of the sensors instead.

6. CONCLUSIONS

The method that has been implemented to resolve the issue of improperly watering plants is a mobile application that can tell its users whether a sample of soil is wet or dry. Recognizing that the soil is dry encourages the users to water their plants, and recognizing that the soil is wet can inform the users that there is no need to water their plants for the time being. By using this application, people can prevent overwatering the plants and accidentally killing them; on the other hand, they can also potentially be alerted to the fact that the plants may not be getting enough water. The application was tested in an experiment in which ten dry soil samples and ten wet soil samples were used to take pictures for the application. The samples were taken outside at various times during the day, which ensured that the soil sample pictures were taken at different lighting levels. The number of times that the application correctly determines the moisture of the soil was recorded in a table separately based on whether the tested soil sample was a wet or dry sample. According to the results, the application's model is very proficient at determining the soil moisture. Because the lighting levels were different across each picture, the model has proven to be somewhat robust across multiple lighting conditions. However, the single inaccuracy of the model from the experiment indicates that while it is not an urgent issue, the model in the application still has room for improvement.

One of the most significant current limiting factors in the application is its ability to use multiple sensors. Currently, only one sensor at a time can be used with the application. However, in a more realistic agricultural setting in which more farmland would have to be analyzed for its soil moisture, the application may be impractical to use [15]. To keep the application relevant within the agricultural field, more time and effort would need to be spent to allow the application to take in multiple sensors at a time and do so in a manner that still keeps the user interface clean and easy to navigate.

Something that could be done is adjusting the sensor page to contain information from multiple sensors. As the current sensor page shows what is being seen by the sensor, the page wouldlikely have to be scrollable so that the user will be able to see a live feed from multiple sensors in one place.

REFERENCES

[1] Schmugge, T. J., Jackson, T. J., & McKim, H. L. (1980). Survey of methods for soil moisture determination. Water Resources Research, 16(6), 961-979.
[2] Mittelbach, H., Lehner, I., & Seneviratne, S. I. (2012). Comparison of four soil moisture sensor types underfield conditions in Switzerland. Journal of Hydrology, 430, 39-49.
[3] Dursun, M., & Ozden, S. (2011). A wireless application of drip irrigation automation supported by soil moisture sensors. Scientific Research and Essays, 6(7), 1573-1582.
[4] Lichtenberg, Erik. "Agriculture and the environment." Handbook of agricultural economics 2 (2002): 1249-1313.
[5] Koop, Steven HA, and Cornelis Johannes van Leeuwen. "The challenges of water, waste and climate change in cities." Environment, development and sustainability 19.2 (2017): 385-418.
[6] Zhao, Cheah Wai, Jayanand Jegatheesan, and Son Chee Loon. "Exploring iot application using raspberry pi." International Journal of Computer Networks and Applications 2.1 (2015): 27-34.
[7] Abadi, Martín. "TensorFlow: learning functions at scale." Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming. 2016.
[8] Schmugge, T. J., T. J. Jackson, and H. L. McKim. "Survey of methods for soil moisture determination." Water Resources Research 16.6 (1980): 961-979.
[9] Morton, T. G., A. J. Gold, and W. M. Sullivan. Influence of overwatering and fertilization on nitrogen losses from home lawns. Vol. 17. No. 1. American Society of Agronomy, Crop Science Society of America, andSoil Science Society of America, 1988.
[10] Bradford, Kent J., and Theodore C. Hsiao. "Stomatal behavior and water relations of waterlogged tomato plants." Plant Physiology 70.5 (1982): 1508-1513.
[11] Moroney, Laurence. "The firebase realtime database." The Definitive Guide to Firebase. Apress, Berkeley, CA, 2017. 51-71.
[12] Islam, Rashedul, Rofiqul Islam, and Tohidul Mazumder. "Mobile application and its global impact." International Journal of Engineering & Technology 10.6 (2010): 72-78.
[13] Pang, Bo, Erik Nijkamp, and Ying Nian Wu. "Deep learning with tensorflow: A review." Journal of Educational and Behavioral Statistics 45.2 (2020): 227-248.
[14] Weiss, Patrice L., et al. "Video capture virtual reality as a flexible and effective rehabilitation tool." Journalof neuroengineering and rehabilitation 1.1 (2004): 1-12.
[15] Ochsner, Tyson E., et al. "State of the art in large-scale soil moisture monitoring." Soil Science Society ofAmerica Journal 77.6 (2013): 1888-1919.