CV Garden

Ryan Kellerman, Ankeet Parikh, Karan Rajput

May 7, 2018

Abstract

Our project focuses on building a robot apparatus that will use computer vision to assist in maintaining a garden. This includes distinguishing between a crop and a weed and different crop types. This is a practical way to decrease amount of manual labor, optimize use of farming resources and reduce costs associated with human error. A core part of the process includes using convolutional neural networks to recognize plants using a mounted camera and sending signals to the apparatus to initiate the appropriate action based on the plant. This report will focus on challenges, configuration choice, results, cost analysis and future work.

1 Introduction

There are many difficulties in maintaining a home garden, such as watering plants on a regular schedule, removing weeds and unwanted material, and identifying any possible infestations. These processes take long manual hours and are prone to errors that vastly decrease the garden's productivity. These problems often lead to gardens infested with weeds and ill cared-for plants.

There have been recent developments in the field of agriculture automation. Some of the projects focus on technology that assist in maintaining a home garden but have inherent shortcomings. Most of the projects that are selling cover a wide problem scope thus are too expensive. Functions include identifying the plant/weed, removing weed, detecting moisture and everything in between. Having all these different functions leads to an expensive apparatus that is hard to maintain.

We will focus on one function: identify the weed/crop. This function is usually enough for a common man who is trying to maintain a small garden. Howevever, we will make the project cohesive so that our project can be easily integrated with others so as to increase the functions that are covered. In this report, we elaborate on a robotic appartus that identifies weeds and crops (and type of) in an effort to assist the caretaker while maintaining the the garden. This will increase garden productivity by identifying weeds before they grow and crops before they die.

2 Objective and Scope

The motivating factor of our project is to integrate concepts of computer vision and machine learning to help in the maintenance of a home garden. To do so, we will make use a convolutional network that can be trained on a variety of plant types and trained to perform well with very minute details. Our plant image classifier works with a large number of images for training. We will be uploading the entire image set as well as our trained network for anyone to use for free. This versatility is very important for the project since there are many types of common garden crops.

Another important part of the project is that our robot is inexpensive to buy and easy to maintain. Our robot will run on an automated car, raspberry pi and a camera. All of these parts are sold inexpensively in the market and have separate functions. When there is a misfunction, the point of error is very obvious and the part is very easy to replace.

Moreover, the code is segmented in such a way that it can be easily imported and integrated with any other project. This is beneficial since it can be added to any apparatus that carries out other functions. In this way, the owner does not have to pay hundreds of extra dollars for functions that are unnecessary.

This project is twofold, in that is has a robotic component as well as a software component. The robotic component consists of a camera mounted on a car which takes pictures of plants. The apparatus is described in more detail later. The software components include all interactions between the raspberry pi and the car, as well as all machine learning and image recognition procedures.

3 Methods

We start by collecting images of 5 different plant types that includes sunflower, daisy, rose, dandelion etc. Several online open source libraries like Kaggle are used to obtain these images which are then used then used to train a neural net. The training dataset consists of about 70 percent of the total images that we have acquired from the open source libraries. Not only is acquiring these images a challenge but it is also tough to assemble the images in a way that we can use it for training our neural network. We will elaborate more on challenges later in the report.

We repeated this procedure with 18 plant types and later, 102 plant types and consistently used 70 percent of the images for training. The accuracy from 5 to 18 plant types stayed about the same but went down for 102 plant types. This is expected since the network has to train on a lot more images and more detail. The network has to handle different plant colors, details, orientations etc.

For training, We use 10 epochs and gained about a 96 percent accuracy with our data. We did not put the images through more epochs since we wanted to avoid over-fitting the data. There are two classification algorithms that work on these images. One classification algorithm is aimed at distinguishing between a crop and a weed. That is the first algorithm that is used in our workflow. The next classification uses the result from the first. If the plant is recognized as a crop, it goes through another classification algorithm which returns the crop type. That crop type is then displayed as the result of the workflow.

We assemble an automated car to serve as a robot on which we can mount our code and camera and classify the crops. The robot is controlled by an arduino board which serves as the power supply for the motors. A shield is used to connect this powering arduino with another arduino board that can be connected to a raspberry pi. The raspberry pi can be thought of as a computer on which we can load our worflow, camera and classification code. The Raspberry Pi features a camera, a microSD card that holds the libraries and a adapter which can be used to integrate with other hardware. We upload both classification algorithms, workflow and camera code to the raspberry pi. The raspberry pi works with the arduino to drive the car, take a picture and return the result back to the user.

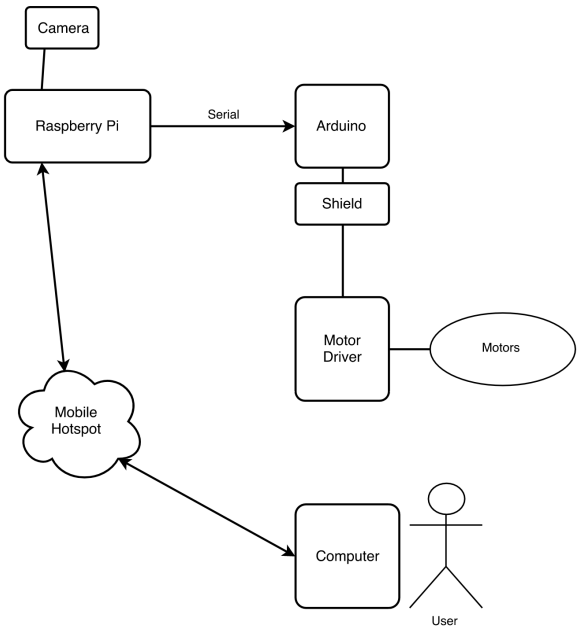

Figure 1: Hardware Setup. The flow starts with the user running the master program on the Raspberry Pi using a mobile hotspot that they are both on. The Raspberry Pi makes a serial connection with the Arduino that is connected to the motor driver with a shield. The car moves forward and when a plant is found, the camera takes a picture. The Raspberry Pi then classifies the picture as crop/weed and crop type (if applicable) and sends the result back to the computer via the same mobile hotspot.

3.1 Technologies

This project combines together technologies from several different realms, including robotics, computer vision, and machine learning. The car we use is an ELGOOG CAR, which comes with an arduino as its onboard processor. The software is loaded onto a Raspberry Pi, which lies on the car alongside the arduino controller. We are using keras primarily to do the image recognition by means of a convolutional neural network. Additionally, Keras uses tensorflow as its back end component. We are also using OpenCV to do some basic image processing. Scikit-learn is also used to help with training the neural network.

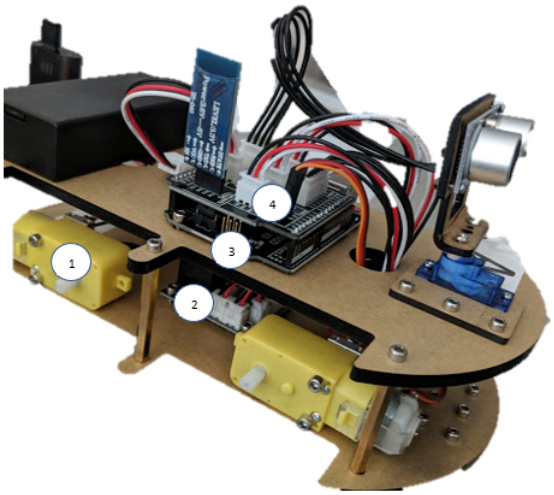

Figure 2: Robot. 1. Motors, 2. Motor Powering Arduino, 3. Arduino to integrate with Raspberry Pi and 4. Shield.

3.2 Collaboration

We tried to get as many images of plants as we could since that helps the program recognize the type with different angles, apparent shape and detail. To get more data, we collaborated with the Rutgers University School of Environmental and Biological Sciences. They supplied a large collection of pictures of cranberries, which we included in our training set.

3.3 Master-Slave Implementation

An important component to our project is the ability to delegate certain tasks to different hardware components. Since the robot kit is geared towards an Arduino controller, our challenge was to find a way to incorporate this into our existing design, which revolved around using the Raspberry Pi as the primary controller. We decided that we could make use of this setup by using a Master-Slave implementation, with the Raspberry Pi acting as the Master, and the Arduino as the slave. Later we will discuss some more details of the procedure and challenges that arose from it, but it is important to note how this delegation of roles is a well known design choice, and that the Master-Slave method extends far beyond just micro-controllers, but can be seen all across electronics, in digital logic, low level computer architecture, and in many other facets of digital design. This design made our system more manageable and helped to establish a more compartmentalized structure.

4 Design

4.1 Neural Network

4.1.1 Construction

We used Keras to build a Convolutional Neural Network (CNN), consisting of several layers. We used 3 convolutional layers, and 2 Dense layers for the network. In addition, we used linear activation func-

tions, as well as a LeakyReLU activation function. We used MaxPooling at each layer, as well as 3x3 convolutions.

For our classifier, we decided to use a Convolutional Neural Network, as other researchers have received promising results using one. We used the Keras machine learning library in Python, as well as Tensorflow for the backend component. Our Neural Network itself consisted of several layers. We used 3 convolutional layers, and 2 dense layers for the network. In addition, we used linear activation functions, as well as a LeakyRelu activation function. We use maxpooling at each layer as well as 3x3 convolutions. It is worth mentioning that the python program that performs the training also has a preprocessing phase where it downsamples the images. We chose (49,49,3) and (90,90,3). [1] [2] [3] [4]

It is important to note that these parameters were not the ones initially chosen, but rather picked after tinkering with the neural network. We had tried changing many different things. For example, we tried using 5x5 convolutions at some of the layers, but found that this did not improve any of the performance. In addition to this, we attempted to change the number of output filters in each convolutional layer, and eventually stuck to 32, 64, and 128 for the respective convolutional layers.

We decided to train with 10 epochs since that gave us an acceptable accuracy without overfitting the data. We were very mindful of training the data properly but wanted to avoid overfitting since that would limit our research. We got the accuracy to increase with each epoch since the images are broken down further and the loss is decresed enormously. We also noticed that the losses are not decreased significantly after 10 epochs which was another indication that we should stick with 10 epochs.

In addition to building our own Neural Network, we made an attempt at transfer learning. Transfer learning can be understood as such: take a pre-trained neural network, freeze some of the layers (do not change their weights), and modify the weights of the rest of the layers through training. This significantly reduces the computational cost of training, especially when the number of layers to be modified is small, and the number of frozen layers is large. Unfortunately, we did not see superior results using this method, so we decided to stick with the neural network that we developed from scratch.

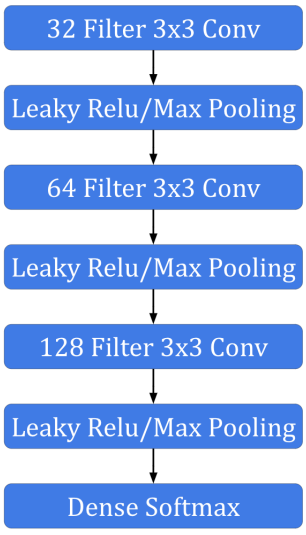

Figure 3: This is the final design of the neural network. Most of this is fairly self-explanatory, and if further elaboration is required, the source code for the construction is shown in the appendix. One thing to note is that the first layer takes in the input shape, so the python program needs to properly resize any image that it would like to classify before the image is fed to the network.

4.1.2 Training

We have collected roughly 30,000 images of plants, including their classes. The plants includes Daisies, Dandelions, Sunflowers, Tulips, and several other classes. For the neural network that would work on 5 classes, We trained on roughly 3,400 images, for 10 epochs. The confusion matrix, precision, recall, and other binary classification measurements, are calculated below.

```
Epoch 2/10
3409/3409 [==============================] - 29s - loss: 0.9908 - acc: 0.6052 - val_loss: 0.9527 - val_acc: 0.6260
Epoch 3/10
3409/3409 [==============================] - 29s - loss: 0.8666 - acc: 0.6641 - val_loss: 0.8967 - val_acc: 0.6530
Epoch 4/10
3409/3409 [==============================] - 29s - loss: 0.7473 - acc: 0.7090 - val_loss: 0.8868 - val_acc: 0.6694
Epoch 5/10
3409/3409 [==============================] - 30s - loss: 0.6241 - acc: 0.7644 - val_loss: 0.8464 - val_acc: 0.6893
Epoch 6/10
3409/3409 [==============================] - 30s - loss: 0.4962 - acc: 0.8102 - val_loss: 0.9392 - val_acc: 0.6565
Epoch 7/10
3409/3409 [==============================] - 28s - loss: 0.3385 - acc: 0.8750 - val_loss: 1.1054 - val_acc: 0.6764
Epoch 8/10
3409/3409 [==============================] - 28s - loss: 0.2091 - acc: 0.9252 - val_loss: 1.1111 - val_acc: 0.6870
Epoch 9/10
3409/3409 [==============================] - 28s - loss: 0.1535 - acc: 0.9534 - val_loss: 1.2850 - val_acc: 0.6846
Epoch 10/10
3409/3409 [==============================] - 28s - loss: 0.1424 - acc: 0.9551 - val_loss: 1.3875 - val_acc: 0.6753
```

Figure 4: Training Procedure. All 3409 images are put through 10 epochs during the training process. Notice that the loss decreases and accuracy increases after each epoch. We chose 10 epochs since we got 95.5 percent accuracy and wanted to avoid overfitting. We also tried to train the data in a reasonable amount of time and anything over 10 epochs would lead to training for over 20 minutes. This number would increase rapidly for more images so 10 epochs is optimal in this case.

	Daisy	Rose	Tulip	Dandelion	Sunflower
Daisy	**122**	20	10	7	14
Rose	28	**124**	1	19	8
Tulip	10	10	**96**	3	42
Dandelion	5	9	3	**132**	7
Sunflower	15	7	41	8	**102**

Figure 5: Confusion Matrix. This matrix shows the hits that each test reported. The first column pertains to hits indicating daisies, second column pertains to hits indicating roses, third column tulips, fourth column dandelion and fifth column sunflower. The first row reports the hits when pictures of daisies are tested, second row reports hits when pictures of sunflowers are tested and so on. The numbers that are positioned on the main diagonal are the true positives, everything else is false positives. Notice that most of our images yielded the right results. Specific numbers are reported below.

	precision	recall	f1-score	support
class 0(Daisy)	0.64	0.71	0.67	173
class 1(Rose)	0.73	0.69	0.71	180
class 2(Tulip)	0.64	0.56	0.60	171
class 3(Dandelion)	0.78	0.85	0.81	156
class 4(Sunflower)	0.59	0.59	0.59	173
avg / total	0.67	0.68	0.67	853

Figure 6: Relevant Measurements. These numbers are derived from the matrix above. Our classification algorithm are very precise and they score highly in precision, recall and f1 score. Pictures of dandelions yielded the best results while sunflower pictures had the most number of false positives. More data for sunflowers will be trained and tested to improve the precision for sunflower pictures.

The results shown above are for 5 classes, but our aim is to expand to more than that. In doing so, we created a classifier that works on 18 classes, and here we show graphs of the loss and the validation accuracy over the training epochs. We chose 10 epochs, as any more seemed to overfit the data. We chose a 70-30 train-test split on 5600 images. (Please note: this is not the same values as shown in the training samples above, as that was for only 5 classes.)

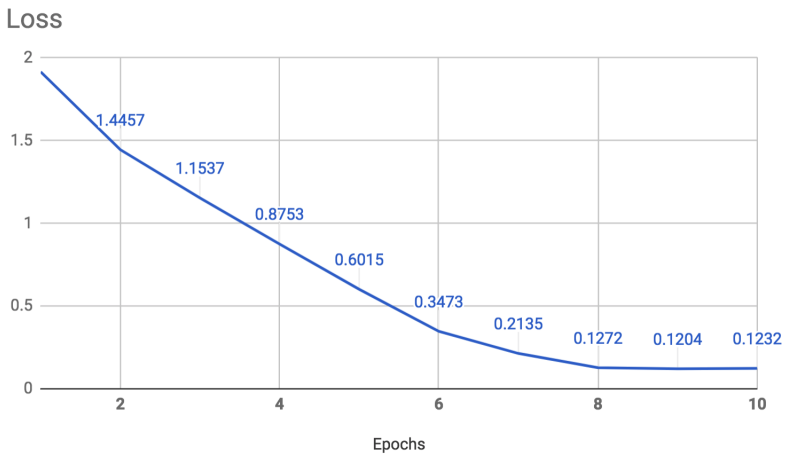

Figure 7: This represents the loss of the 18-class neural network. Simply put, loss measures the difference between the output of the neural network, and the actual output. The objective is to minimize loss, and we can see that the neural network achieves this through optimizing its weights after each epoch.

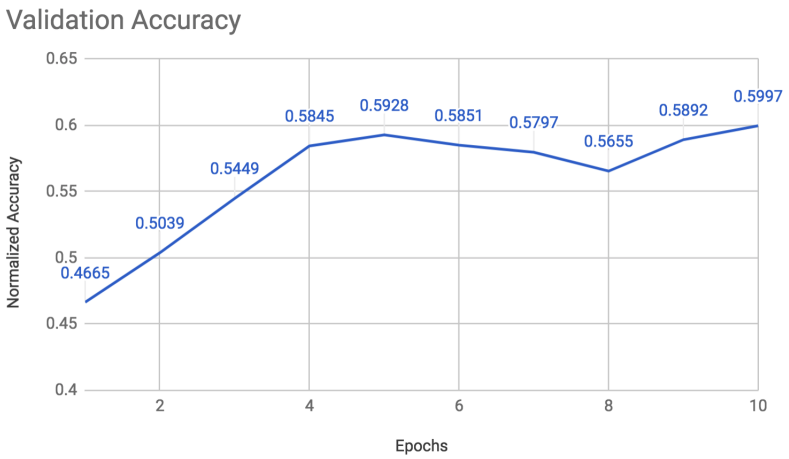

Figure 8: This figure represents the accuracy of the 18 class neural network. The validation accuracy is a good indicator of the performance of the neural network on real life data, which would tell you how well your network is performing. We can see that the final accuracy after 10 epochs hovers slightly under 60%.

5 Apparatus

In order to properly maintain the garden, we need a method to take pictures of the garden. Currently, we have

5.1 Image Recognition Logic

Here is the flow for recognizing an image:

1. Move car to intended location.
2. Stop car completely: this step is needed to ensure a proper image is taken.
3. Take image from mounted camera, and send it to the raspberry pi.
4. Load the image into the program, reshape image as necessary, and load the neural network.

5. Use the neural network to determine the probabilities that the image belongs to a specific class. Note that the probabilities, over all classes, sum to one.
6. Choose the class with the greatest probability as the plant type, print it to the screen and quit the program.
7. Permit the car to move again

Algorithm 1 Master Program

```
1:  procedure MAIN
2:      arduino ← initialize arduino library
3:      camera ← initialize camera interface
4:      network ← load neural network model
5:      while true do
6:          arduino.moveCarForward()
7:          image ← camera.captureImage()
8:          image ← preprocessImage(image)
9:          isCrop ← network.ClassifyIsCrop(image)
10:         if isCrop then
11:             cropType ← network.ClassifyCropType(image)
12:             *Report crop type*
13:             continue
```

Figure 9: Pseudocode depicting the general structure of the master program that runs on the raspberry pi. The program is written in python, and uses imported deep learning and Arduino libraries to accomplish all the tasks necessary. The architecture of the code was written to be very simple and compartmentalized. The process of writing the code involved first writing a very basic skeleton program with empty functions. We then filled in each function one at a time, where each function was a basic task of the system. The result is a main function that is easy to understand and edit when necessary.

6 Cost Analysis

The costs associated with our project encompasses the cost of the car, Raspberry Pi and plants. Below is the breakdown of the cost of our project:

Item	Cost
Car	50
Raspberry Pi	36
Camera	27
Plants	30
Total	143

Table 1: This is the breakdown of the costs associated with the supplies that are used in CV Garden. Notice that the supplies are relatively low in price. This is great news for anyone that wants to use our apparatus in their garden or even integrate it with their machines - they will be able to do so at a relatively low price.

Notice that the cost of the car is much lower than other similar projects. We can further lower the cost of the robot by making our own mount or picking a different automated car. Furthermore, the cost of the plants is just for our overall project, and not to be included in the cost of the car. We will further decrease the cost by buying the Raspberry Pi and camera in bulk so we will get the items at a discounted price per item. Even if the car is sold at a profitable margin, the price of the robot is much more affortable than robots that are currently available in the market. This satisfies one of the main objectives of our project. We want to make a product that can be easily distributed in houses and this cost analysis proves exactly that.

7 Discussion

The data shown in the section 5 provides a few interesting results which we will explain here. First, the following equations show what constitutes some of the benchmarks of the performance of a neural network.

TP = Number of True Positives	$\text{Precision} = \frac{TP}{TP+FP}$
FP = Number of False Positives	$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN}$
TN = Number of True Negatives	$\text{Recall} = \frac{TP}{TP+FN}$
FN = Number of False Negatives	$\text{F1 Score} = \frac{2TP}{2TP+FP+FN}$

First, note that the accuracy, roughly 95.5% is quite high. It might seem natural to claim that the neural network is sufficiently good, but high accuracy can be misleading. Accuracy answers the question "how often do you get the answer right?". The reason this value is misleading is that a classifier that determines whether or not a car is being robbed is likely to say 'NO' quite often, since it is not very likely that a car alarm is set off by a robber every time. However, if the classifier says 'NO' when the answer is 'YES', then the mistake is considered quite significant, and hence the classifier may not be considered that good, despite its high accuracy. It is necessary to consider the set of measurements as a whole, because any single one of them can be biased to become arbitrarily high. Our network presents us with a precision score of 0.67, which answers the question "when the classifier says 'YES', how often is it right?". Recall answers the question "when the correct answer is 'YES', how often does the classifier output 'YES'"? The F1 score is the harmonic average of precision and recall. The rest of our values hover around 70%, which indicates that the classifier is genuinely doing a good job in recognizing the right types of plants.

Looking at the confusion matrix, most of the entries are on the diagonal, which is what should happen in the ideal case. However, because our network is not perfect, there are several things to consider. The last column has 42 entries in the incorrect position. This means that $42/173 = 24\%$ of the sunflowers were classified as tulips. We will soon investigate ways to reduce this value and others like it.

8 Challenges

There were several obstacles that we faced in order to get the skeleton of this project together.

8.1 Acquiring data

Acquiring lots of images of every crop type is important to train our machine learning algorithm. However, it was hard to find images that we could use for our classifier. However, finding useful images was really tough to do. We could not find images in the right format, with a good background, correct orientation etc. Even when we obtained the dataset for 102 plant types, the images were not ordered in any particular way. The plants were given numbers that pertained to a plant type but the plant type each number corresponded with was not reported. We would also need to modify the image formats into something that we could use, which required writing some python scripts.

8.2 Automated Car

After we built the car, we realized that the L298N motor driver board that controls the car does not work properly. It took us time to figure out that while the left side of the board (that controls the left wheels) worked as intended, the right side did not propel the right wheels' motors. Hence, the car just went around in circles. So, we edited the code to assign one port of the left side of the board to control one of the right wheels on the car. This makes the car go forward in a straight line.

8.3 Neural Net Performance

With any neural network, the issue of accuracy/recall/precision/etc. is of major importance. We followed a standard framework for recognizing plants, but when we tried to tweak the parameters, add layers, and tinker with the convolutional layers, we ran into problems with improving the measurements of our classifier. We tried adding more convolutional layers, changing the 2d convolution size from 3x3 to 5x5, testing the activation functions from softmax, linear, LeakyReLU. Eventually, we settled on the final design as described above.

Additionally, we had trouble when attempting to modify the neural network to work on 17 plant types when it was initially designed for only 5 of them. At first, we thought to simply increase the number of output nodes accordingly, but doing so unfortunately lead to quite poor results. Our next idea was to increase the number of layers, for an increased complexity might allow the neural network to perform more sophisticated feature extraction. However, such a simple transformation did not lead to improved results. In order to genuinely do something useful, we needed to take a step back, go on the internet, and do some more research in the area. We read several papers, which helped guide us in constructing a network with better performance.

8.4 Linking Arduino to Raspberry Pi

The software that allowed us to control the Arduino with the Raspberry Pi was very clever. The basis of this connection was a Master-Slave implementation, with the Raspberry Pi being the "master" and the Arduino as the "slave." In order to accomplish this, we downloaded a library called nanpy, which included both Arduino and Python software. The Arduino code essentially used the serial port to listen for requests from the "master" nanpy software, which were then processed into functions that the Arduino could understand. In the python code, we were able to use several simple function calls that established a connection through the serial (USB) port, and then sent commands to the Arduino through the use of python friendly commands. In this way, it was only necessary to write the logic for the Arduino on the Raspberry Pi end. A problem that arose from this implementation is that after a serial connection has been established between these two devices, the connection was getting lost and an error was thrown upon connection the shield to the Arduino board. Our initial inclination was that this dropped connection was a result of a lack of power necessary to drive all the connected hardware, but we were able to power the Arduino with an external high power supply with no success. We investigated new software as well, but no library or support package was able to salvage the connection. The solution to the problem was removing the connection between the shield and the RX and TX pins on the Arduino board.

8.5 Loading libraries on Raspberry Pi

The Raspberry Pi holds the master code from which the all the required libraries are imported and the car is controlled (since it communicates with the Arduino). However, Rutgers wifi does not have a driver for Raspberry Pi and our mobile hotspots are not fast enough for the libraries to install without getting timed out first. So, we had to take the Raspberry Pi home to download the libraries. The process of finding which libraries to install was also a tedious task. The greatest challenge was installing dependencies that are compatible with Python 2.7, and further, compatible with each other. We thus found that using pip2 and apt-get to install libraries were the only methods that completed successfully. Libraries for python 3.6 were typically unable to build. After much trial and error, we were able to successfully install all dependencies needed to support keras.

8.6 Camera

The Raspberry Pi camera module is an inexpensive, versatile tool for simple application. However, it is susceptible to wear and tear, and we first began experiencing trouble capturing images, and then connecting to the camera sensor entirely. We believe the cause was connecting the device while the raspberry pi was powered on (which is never advised), or from an exposed connection shorting the device. The end result was several hours spent debugging and eventually looking towards purchasing a new camera module.

8.7 Taking Live Photos

The images that the neural net was trained on were fairly consistent in background, color and detail. The neural network's accuracy was affected while taking live photos, however, since taking pictures imposes several challenges. One challenge is the orientation of the plant when the picture is taken. The orientation affects the detail that the neural network recognizes hence might affect the classification. Moreover, the background of the live pictures can easily affect the classification. The images that we found on Kaggle and other sources made a point to have backgrounds that differ from the plant color so the detail can be easily detected. However, while taking photos, the background can make detecting the plant much more difficult, especially if there are other plants in the background. These factors will make the neural network less accurate than what we aimed for.

9 Future Work

In the short term, there are several goals that our team would like to accomplish. First, we would like to see greater improvements to the neural network. These improvements can come with a neural network that has better indexed and can parse images better. It will also be better if we trained our network progressively. For example, we jumped from 5 plant types to 18 plant types to 102 plant types. If we trained on five types then six types then seven and so on, we could have pin pointed exactly at which point the neural network starts to decrease in accuracy.

We would also like to test the car more on actual plants rather than pictures of plants as we started to do. Taking the time constraint associated with this project, we did not have much time to test out the neural network's performance with actual plants. We have done extensive testing on 2D plant images and we got great results. But, the product is meant for 3D images so testing on that would be much more applicable and useful. There are problems that can come up with live pictures like backgrounds, blurry pictures, smothered detail etc. It would help to pinpoint the shortcomings that are associated with taking pictures of live photos.

Additionally, it is possible that we can determine if plants are dead or if they are a victim of a fungal infestation (for example). Incorporating this aspect will further increase garden productivity since infestations can spread very quickly and kill other healthy crops. Computer vision can be done to do this but it can be tricky since it will add another aspect to our weed/crop classifications right now. We will also need to find datasets that have plant diseases which might be very difficult to find.

There are a wide range of possibilities for the long term future of this project. Our network can possibly be mounted on the top of a drone to take pictures from a greater height. Another possibility is that the network can be greatly expanded, modularized, and changed to identify different classes of plants. For example, we can have an initial neural network that determines whether the plant is a seedling or is mature, and then subsequently loads a second neural network depending on whether the plant is mature or is a seedling. The second neural network then proceeds to do a species-level classification.

Another extension of this project is to make an Android/IoS application that interacts with our robot apparatus. The application can report plant diseases, watering schedules, weeds growing etc. This would allow the user to remotely keep track of what is happening in their garden. The app can also keep inventory of gardening supplies and remind user to pick up fertilizer or plant food whenever needed. This would work especially well for everyone especially working people who cannot always tend to their gardens especially if they spend extended hours away from home or are traveling.

Our original design had some very ambitious plans for the functionality of the robot. Some things we would like to have seen was the incorporation of a moisture sensor and a water dripper to automatically handle watering schedules. Another large component of potential work is distinguishing between crops and weeds. Doing so would require a large dataset of weed images to train on. The difficulty with this task is that while there are images of mature weeds, there are very few research images for young weeds, which is what one would typically find in a garden. After identifying the weed, the system would be able to notify the user, or even make use of a robotic weed picker. A final idea is after identifying the plant type, the system would access a local database or a web API to get specific care instructions, such as branch maintenance, sunlight needs, etc. These details would make for a comprehensive system that would truly make gardening a hands-off task.

10 Current Trends in Robotics

Similar projects are currently being developed in the market. One such project is Farmbot[5]. It uses CNC machine that uses special tools and software to grow plants. They employ linear guides to precisely place seed injectors, watering nozzles, sensors etc. They also have a web application that the user can use to graphically design their garden to their desired specifications and synchronize. However, one of the biggest problems with the project is that it costs anywhere around four thousand dollars to install. For a common person trying to maintain a garden, this is quite a deep hole in their pockets especially considering that the machine will require regular maintenance, which is also not cheap. This is where the CV Garden is especially useful. It is made of relatively cheap and easily replaceable parts so it neither costs a lot to buy nor is it expensive to maintain. This enables someone to easily get help in their garden rather than lose a lot of money over an apparatus that will require a lot of maintenance anyway. Such a project is more geared towards farmers who make a business out of their agriculture, rather than ones who do it as a hobby.

TeraSentia[6] is another such project developed by students at University of Illinois at Urbana Champaign. It is an agricultural robot which autonomously measures crop traits to reduce manual labor in the field. Their goal is to help scientists and farmers work with more accurate and easily gathered data that they would otherwise have to spend days working towards. The robot comes with two visual cameras, a tablet app with first person view and cloud software to store data on and train the robot. The robot currently counts plants and measures stem width to help estimate biomass. More work is done to teach the robot other skills like measuring corn ear height, angle plant height and recognizing diseases. This is another project that is geared at farmers that deal with acres of plants and can afford to buy a robot that costs five thousand dollars just to count plants and measure stem width.

The MIT open agriculture[7] initiative is a project that uses computer vision and machine learning techniques for the classification and identification of plants. The intended goal here is to construct an open-source toolkit for plant analysis, that can be used by anyone, whether it be a company tending to a large farm, or someone working on a smaller, home garden. Currently, the method works as such: a plant is placed so that the camera gets a bird's eye view of the plant, and then, in real time, the vision libraries detect the plant, count the number of leaves, and measures the dimensions of a bounding box. Additionally, researchers at MIT has developed something called a *Personal Food Computer* [8] which includes pH and CO_2 sensors, as well as a camera that takes pictures of the growing plants. We believe that our progress in computer vision can help aid projects like these with monitoring the growth and development of the garden.

Another current product in the field is a weeding robot from Franklin Robotics called Tertill[9]. Tertill is solar powered, and it physically moves along the garden, and when it comes across a weed, it plucks it out using a set of pickers. It moves in a manner similar to a roomba. Though the robotic component of the Tertill is promising, the issue is with how it distinguishes between weeds and crops: crops are tall and weeds are short. Short crops are surrounded by tall wire so that they are not plucked. The main issue with this system is that there is no way to pluck crops out if they have gone bad (wilted leaves, infestation of fungus, etc.). We believe that our project allows for improvements to this project by providing the ability to recognize plants more effectively rather than simply differentiating based on height.

A project that has similar intentions as ours is the Plant Classification System for Weed/Crop Discrimination without Segmentation [10]. In this project, the research team decided to create their own data set by mounting a camera to an autonomous field robot, capturing images in the visible red (R) and near infrared (NIR) spectrum. The images that they captured are unique; they contain several different plants and weeds in each photo. The objective was to be able to classify and identify among all the types of plants found in the images. The images they collected were pre-processed to remove the soil in the background, as well as isolate the shape of the vegetation and emphasize features of the image that could be extracted more easily by a training algorithm. The results are also post-processed via spatial smoothing and interpolation, a process that led to even further performance improvements to the classification process. The results of this research project are quite impressive, as the system is able to—very accurately—produce an image that has each plant in the image color coded to correspond to either a crop or a weed, and further, the type of crop if found to be a crop.

There is also a more class of robots being developed called agriculture robots, or agbots[11]. These robots are aimed at monitoring soil moisture, water levels and collecting data that can be stored and analyzed. There are advanced agbots that are being developed to detect changes happening in one plant of a plant type but not to another plant of the same type. Such robots promise more flexibility for the common person but as other projects, they cost a lot of money.

In addition to hardware, we would like to mention some work being done in the area of plant and flower classification. [12] The ResNet [13] architecture is considered state of the art in this domain. ResNet, short for Residual neural network, uses a concept called *shortcut connections*, which skip one or more layers in the neural network. The authors of ResNet hypothesized that residual mappings are easier to optimize than regular ones. This greatly improves the performance of the neural network. In addition to ResNet, there are several others, like AlexNet and SqueezeNet. SqueezeNet attempts to reduce the computational figerprint of a CNN by making use of compression techniques.

While it is true that such projects promise to improve productivity in the field, it cannot be ignored that as robots take over agriculture, people's jobs are in danger. There are millions of workers on the farm that carry out the current manual processes that are associated with the farm. For many, the introduction of robots in agriculture will mean that there is no longer a place for them in the industry. Another disadvantage with robots is that while they are mostly accurate, they are bound to make mistakes. For robots that are still in training, this can mean high probabilities of mistakes and losses to the owner. While robots offer a lot of benefits, one has to consider the downsides that come with them and balance them to find the the optimal fit between using a machine and a human hand in the field.

In conclusion, the field of automated farming is a rapidly developing one, due to considerations of population, issues in farming, food preservation, and costs. Technology can help mitigate many of the struggles of farming, and in particualar, issues of weeds affecting the crop. We believe that our use of computer vision in plant recognition can greatly help the industry in performing tasks regarding the identification of species (invasive or otherwise).

References

[1] C. O. Neal, "Crops vs. weeds." [Online]. Available: https://www.kaggle.com/limitpointinf0/crop-vs-weeds

[2] S. G. Wu, F. S. Bao, E. Y. Xu, Y. Wang, Y. Chang, and Q. Xiang, "A leaf recognition algorithm for plant classification using probabilistic neural network," *CoRR*, vol. abs/0707.4289, 2007. [Online]. Available: http://arxiv.org/abs/0707.4289

[3] A. Gurnani and V. Mavani, "Flower categorization using deep convolutional neural networks," *CoRR*, vol. abs/1708.03763, 2017. [Online]. Available: http://arxiv.org/abs/1708.03763

[4] D. Guru, Y. S. Kumar, and S. Manjunath, "Textural features in flower classification," *Mathematical and Computer Modelling*, vol. 54, no. 3, pp. 1030 – 1036, 2011, mathematical and Computer Modeling in agriculture (CCTA 2010). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0895717710005236

[5] "Farmbot: Open source cnc farming." [Online]. Available: https://farm.bot/

[6] "Terrasentia." [Online]. Available: https://www.earthsense.co/home/

[7] "Mit open agriculture initiative: Computer vision and machine learning." [Online]. Available: https://www.media.mit.edu/projects/food-computer-project-open-agriculture-initiative/overview/

[8] E. C. Ferrer, J. Rye, G. Brander, T. Savas, D. Chambers, H. England, and C. Harper, "Personal food computer: A new device for controlled-environment agriculture," *CoRR*, vol. abs/1706.05104, 2017. [Online]. Available: http://arxiv.org/abs/1706.05104

[9] "Tertill: The solar powered weeding robot for home gardens." [Online]. Available: https://www.kickstarter.com/projects/rorymackean/tertill-the-solar-powered-weeding-robot-for-home-g

[10] "Plant classification system for crop / weed discrimination without segmentation." [Online]. Available: https://pdfs.semanticscholar.org/8483/43ff72682995da16e18b56d54823ab177474.pdf

[11] "Agbot challenge." [Online]. Available: http://www.agbot.ag/

[12] J. Waldchen, M. Rzanny, M. Seeland, and P. Mader, "Automated plant species identification-trends and future directions," *PLOS*, 2018. [Online]. Available: http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1005993

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

11 Appendix

All source code with comments can be found on our GitHub page.

11.1 Code for neural network

```
import os
from PIL import Image, ImageFile
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import keras
from keras.models import Sequential,Input,Model,load_model
from keras.layers import Dense,Dropout,Flatten
from keras.layers import Conv2D,MaxPooling2D
from keras.layers.normalization import BatchNormalization
from keras.layers.advanced_activations import LeakyReLU
from keras.applications.vgg16 import VGG16, preprocess_input
import scipy.misc
from skimage import transform
import warnings
import cv2

from keras.callbacks import History
history = History()
from keras.utils import np_utils

ImageFile.LOAD_TRUNCATED_IMAGES = True
warnings.filterwarnings("ignore")

train_dir = "./input/train102";
train_list = os.listdir(train_dir);
records = [];

for category in train_list:
img_list = os.listdir(train_dir + "/" + category);
for img in img_list:
records.append((img, category));

print(len(train_list))
df_train = pd.DataFrame.from_records(records, columns = ['image', 'category']);

print(df_train.head());


dim_image = [];
for i in (train_dir + "/" + df_train["category"] + "/" + df_train["image"]):
img = Image.open(i);
data = img.size;
dim_image.append(data[0]);

# print('smallest image dimension', min(dim_image))

i_height = min(dim_image);
i_width = min(dim_image);

X = []
count = 0
```

```
bad_images = []
for i in (train_dir + "/" + df_train["category"] + "/" + df_train["image"]):
img = Image.open(i);
img.load();
img = np.asarray(img, dtype = "float32");
img = img/255;
data = transform.resize(img, (90, 90))
if data.size != 90*90*3:
bad_images.append(count);
count = count + 1;



df_train = df_train.drop(df_train.index[bad_images]);

for i in (train_dir + "/" + df_train['category'] + '/' + df_train['image']):
img = Image.open(i).convert('RGB')
img = np.asarray(img, dtype='float32')
img = img/255
data = transform.resize(img,(90,90))
X.append(data)

X = np.array(X)

y = np.array(df_train['category'].astype('category').cat.codes)

# print(X)
# print(y)
print('Done creating X and y.')
print('X Shape',X.shape)
print('Y Shape',y.shape)

test_dir = './input/test/'
test_list = os.listdir(test_dir)
print('Train Data', len(df_train.index))
print('Test Data',type(test_list),len(test_list))
print('categories',os.listdir(train_dir))
print('# of categories', len(os.listdir(train_dir)))
im_shape = (90, 90, 3)
batch_size = 10
nb_epoch = 20
nb_classes = len(train_list)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=42)
Y_train = np_utils.to_categorical(y_train, nb_classes)
Y_test = np_utils.to_categorical(y_test, nb_classes)

cnn = Sequential([
Conv2D(32, kernel_size=(3,3), activation="linear", input_shape=im_shape, padding="same"),
LeakyReLU(alpha=0.1),
MaxPooling2D((2,2), padding="same"),
Conv2D(64, kernel_size=(3,3), activation="linear", padding="same"),
LeakyReLU(alpha = 0.1),
MaxPooling2D((2,2), padding="same"),
Conv2D(128, kernel_size=(3,3), activation="linear", padding="same"),
LeakyReLU(alpha=0.1),
MaxPooling2D((2,2), padding="same"),
Flatten(),
```

```
Dense(50, activation="relu"),
Dense(len(train_list), activation="softmax")
])

cnn.compile(loss="sparse_categorical_crossentropy", optimizer=keras.optimizers.Adam(lr=0.001), metric
hist = cnn.fit(X_train, y_train, batch_size=batch_size, epochs=nb_epoch, verbose = 1, validation_data

cnn.save('network.h5');

cnn = load_model('network.h5');
images = ["0a64e3e6c.png", "0ad9e7dfb.png", "0ae6668fa.png"]; # sample images

for st in images:
im = Image.open("./input/test/" + st).convert("RGB");
im.load();
im = np.asarray(im, dtype = "float32")
im = im/255
im = transform.resize(im, (49, 49));
check = np.array([im])
prediction = cnn.predict(check)
print(st)
print(prediction[0])

print(train_list)


# visualizing losses and data accuracy
train_loss=hist.history['loss']
val_loss=hist.history['val_loss']
train_acc=hist.history['acc']
val_acc=hist.history['val_acc']
xc=range(nb_epoch)

plt.figure(1,figsize=(7,5))
plt.plot(xc,train_loss)
plt.plot(xc,val_loss)
plt.xlabel('num of Epochs')
plt.ylabel('loss')
plt.title('train_loss vs val_loss')
plt.grid(True)
plt.legend(['train','val'])
print plt.style.available # use bmh, classic,ggplot for big pictures
plt.style.use(['classic'])

plt.figure(2,figsize=(7,5))
plt.plot(xc,train_acc)
plt.plot(xc,val_acc)
plt.xlabel('num of Epochs')
plt.ylabel('accuracy')
plt.title('train_acc vs val_acc')
plt.grid(True)
plt.legend(['train','val'],loc=4)
#print plt.style.available # use bmh, classic,ggplot for big pictures
plt.style.use(['classic'])


# confusion matrix
from sklearn.metrics import classification_report,confusion_matrix
```

```
Y_pred = cnn.predict(X_test)
print(Y_pred)
y_pred = cnn.predict_classes(X_test)
print(y_pred)

p = cnn.predict_proba(X_test) # to predict probability
print("")
target_names = ['class 0(Pansy)', 'class 1(Daffodil)']
print(classification_report(np.argmax(Y_test,axis=1), y_pred,target_names=target_names))
print(confusion_matrix(np.argmax(Y_test,axis=1), y_pred))
```