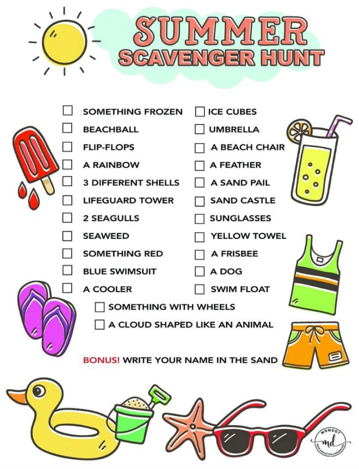
Scavenger Hunt

Marjoree Fargas & Kara Newman

Concept

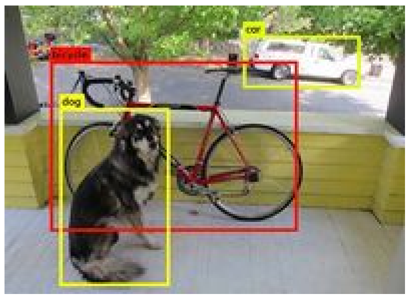
The concept of Scavenger Hunt was to implement the idea of machine learning into the realm of technology and innocence associated with learning primary tasks and languages. This is similarly seen in children or in individuals learning a new language and basic functions. Our project includes a Raspberry Pi camera attached to an Arduino unit. We were inspired by the material in the course as well as the different concepts of machine learning we found through our research. We intended on having our robot unit recognize an object, move towards it and take a picture to "collect" it as a completed task on the scavenger hunt. In order to complete the hunt, the robot must be able to recognize each object from different angles as it moves around the room. We want to create a list of objects for it to seek and complete as it goes on its "hunt" as it would indicate an understanding of its task and ability to execute it. We intended on teaching the Raspberry Pi how to detect the objects through a machine learning tutorial and collaborate with the Arduino's robotic movement code to create a system of moving, searching, and identifying. This is to mimic mannerisms of a child learning to identify an object and actively seek them out as though it were a scavenger hunt. We additionally aimed to design a set and costume for the robot to enhance this effect.



SUMMER
SCAVENGER HUNT

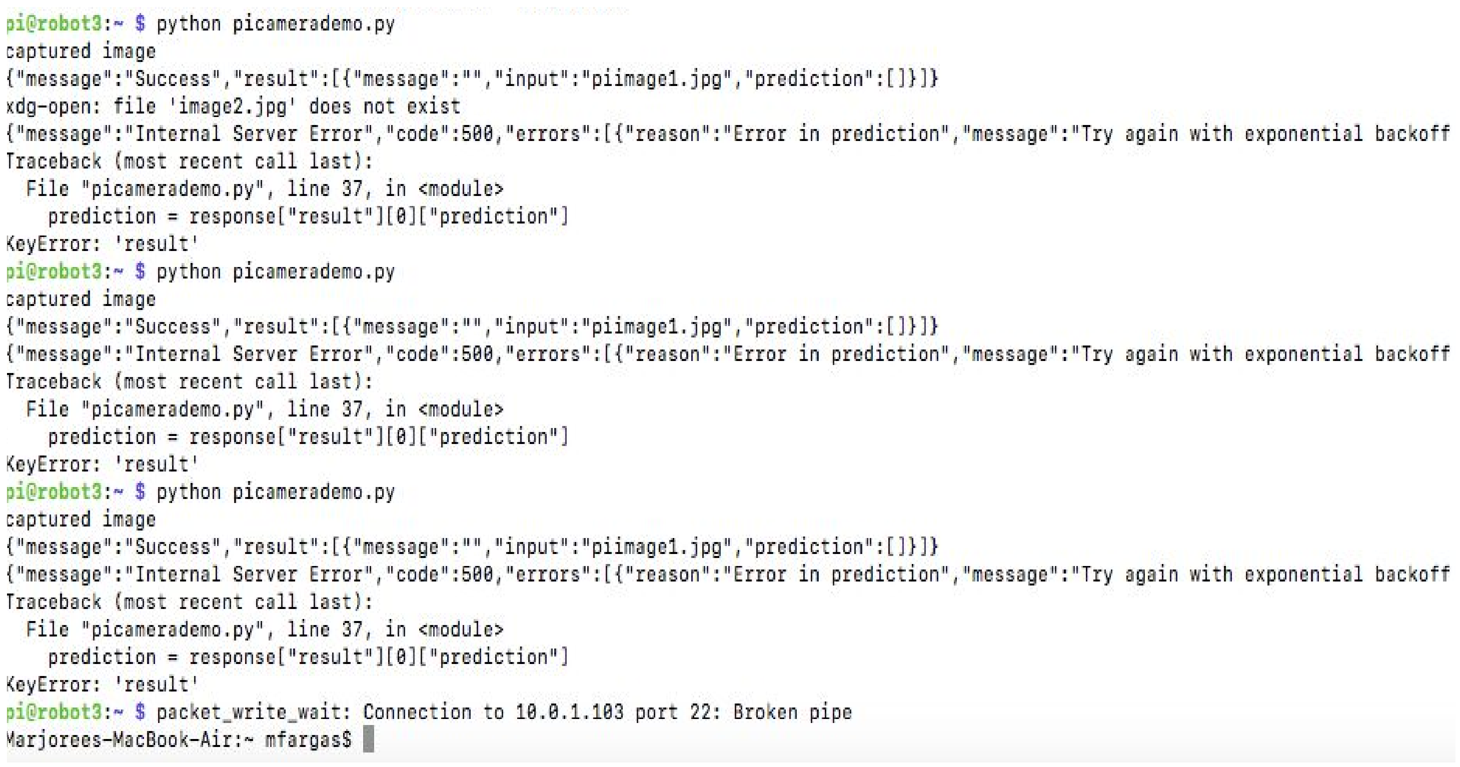
- ☐ SOMETHING FROZEN
- ☐ ICE CUBES
- ☐ BEACHBALL
- ☐ UMBRELLA
- ☐ FLIP-FLOPS
- ☐ A BEACH CHAIR
- ☐ A RAINBOW
- ☐ A FEATHER
- ☐ 3 DIFFERENT SHELLS
- ☐ A SAND PAIL
- ☐ LIFEGUARD TOWER
- ☐ SAND CASTLE
- ☐ 2 SEAGULLS
- ☐ SUNGLASSES
- ☐ SEAWEED
- ☐ YELLOW TOWEL
- ☐ SOMETHING RED
- ☐ A FRISBEE
- ☐ BLUE SWIMSUIT
- ☐ A DOG
- ☐ A COOLER
- ☐ SWIM FLOAT
- ☐ SOMETHING WITH WHEELS
- ☐ A CLOUD SHAPED LIKE AN ANIMAL

BONUS! WRITE YOUR NAME IN THE SAND



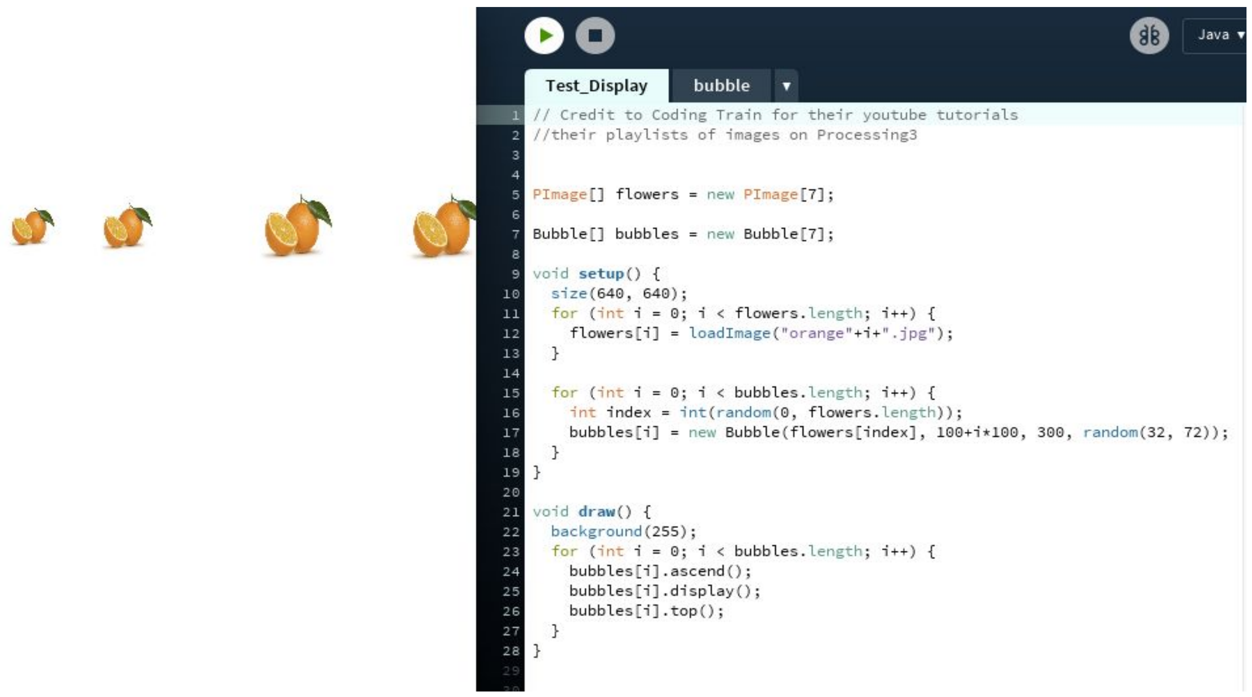
Implementation

We used a tutorial we found on the Medium's article on deep learning with Raspberry Pi. Through this method, we used the Nanonets system of creating a model and obtaining an API key. The API key and model numbers are essential for the url to indicate which model it will access within its system and authorize the use of this model. Our model consists of five different categories of objects—pinecones, tents, water bottles, oranges, and hats—with each category containing a set number of annotated images to learn from. We gathered between 50-100 images per category from various angles to simulate a sense of "learning" from the model. When presented with an object similar to the summarized interpretation of those collections of annotated images, it will draw a rectangle around the object and categorize it. This model was trained using Anaconda and implemented on the Raspberry Pi before introducing movement. With the Arduino part, the robot incorporated robotic movements with its search to simulate a scavenger hunt. We manipulated code from the course GitHub to simulate movement and allow the robot to stop when it sensed an object or boundary.



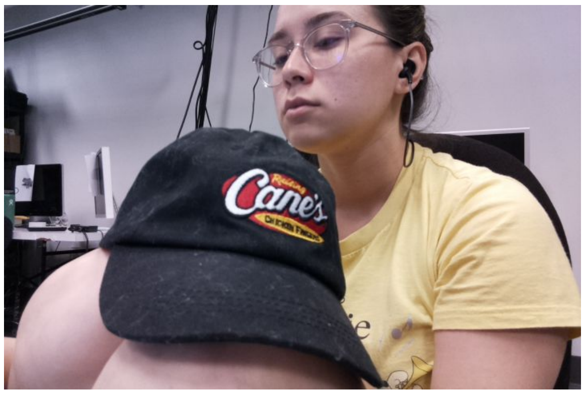
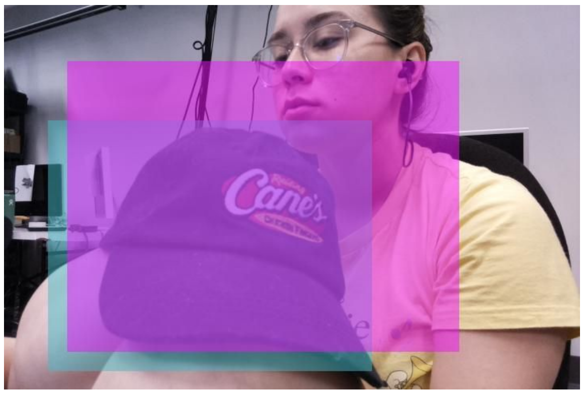
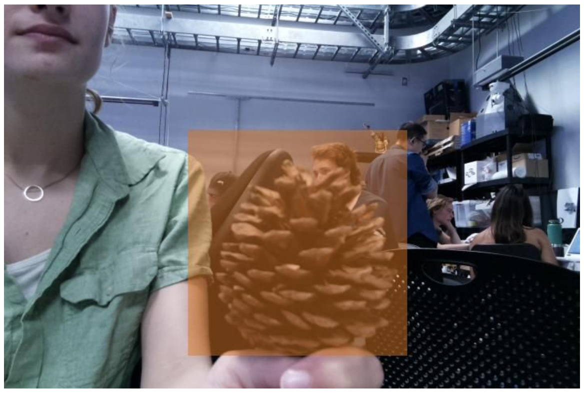
```
pi@robot3:~ $ python picamerademo.py
captured image
{"message":"Success","result":[{"message":"","input":"piimage1.jpg","prediction":[]}]}
xdg-open: file 'image2.jpg' does not exist
{"message":"Internal Server Error","code":500,"errors":[{"reason":"Error in prediction","message":"Try again with exponential backoff
Traceback (most recent call last):
  File "picamerademo.py", line 37, in <module>
    prediction = response["result"][0]["prediction"]
KeyError: 'result'
pi@robot3:~ $ python picamerademo.py
captured image
{"message":"Success","result":[{"message":"","input":"piimage1.jpg","prediction":[]}]}
{"message":"Internal Server Error","code":500,"errors":[{"reason":"Error in prediction","message":"Try again with exponential backoff
Traceback (most recent call last):
  File "picamerademo.py", line 37, in <module>
    prediction = response["result"][0]["prediction"]
KeyError: 'result'
pi@robot3:~ $ python picamerademo.py
captured image
{"message":"Success","result":[{"message":"","input":"piimage1.jpg","prediction":[]}]}
{"message":"Internal Server Error","code":500,"errors":[{"reason":"Error in prediction","message":"Try again with exponential backoff
Traceback (most recent call last):
  File "picamerademo.py", line 37, in <module>
    prediction = response["result"][0]["prediction"]
KeyError: 'result'
pi@robot3:~ $ packet_write_wait: Connection to 10.0.1.103 port 22: Broken pipe
Marjorees-MacBook-Air:~ mfargas$
```

- ➢ Above is an image indicating the success of the recognition of objects, however at this stage of the process we were working on the coordination with our results. This would also be prior to our inclusion of its display on Processing3.



```java
// Credit to Coding Train for their youtube tutorials
//their playlists of images on Processing3


PImage[] flowers = new PImage[7];

Bubble[] bubbles = new Bubble[7];

void setup() {
  size(640, 640);
  for (int i = 0; i < flowers.length; i++) {
    flowers[i] = loadImage("orange"+i+".jpg");
  }

  for (int i = 0; i < bubbles.length; i++) {
    int index = int(random(0, flowers.length));
    bubbles[i] = new Bubble(flowers[index], 100+i*100, 300, random(32, 72));
  }
}

void draw() {
  background(255);
  for (int i = 0; i < bubbles.length; i++) {
    bubbles[i].ascend();
    bubbles[i].display();
    bubbles[i].top();
  }
}
```

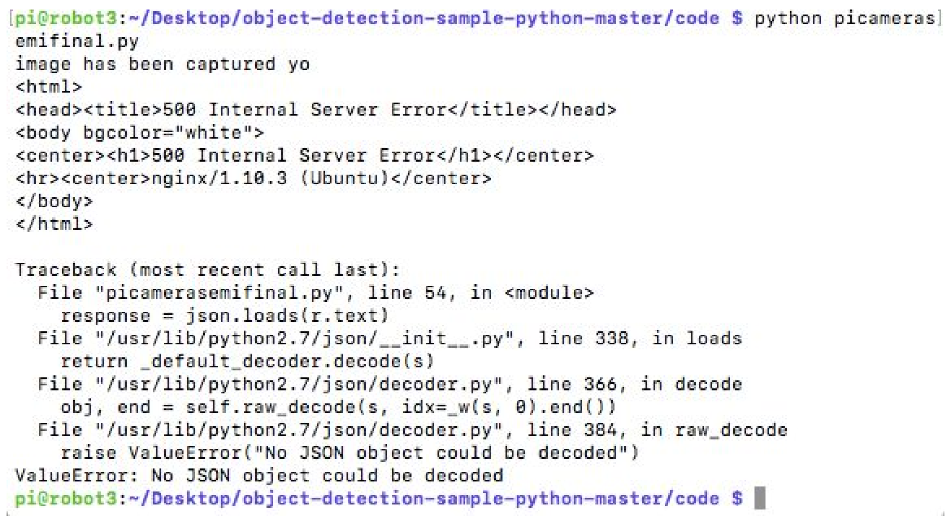
➢ Our images were projected onto a Processing3 Java sketch as our collected images taken and annotated by the robot will be saved into an "images" folder. From that folder, the Java sketch will extract the images and project them onto the screen.

➢ Results gained when testing the object recognition on the Raspberry pi. Hats and pinecones are two of five categories our model has been trained to learn and recognize.

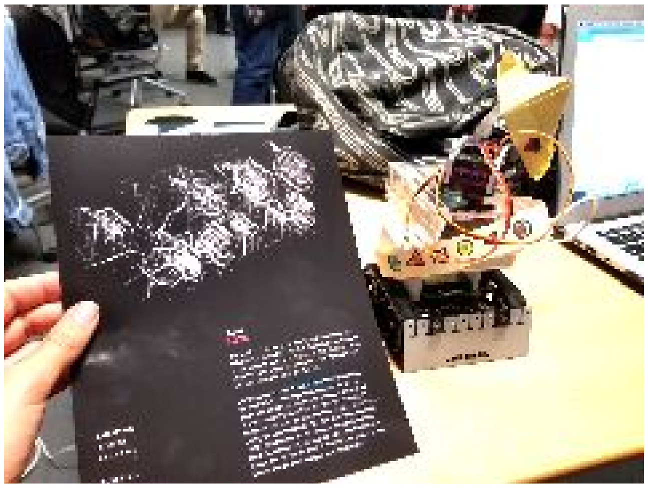
Final Result

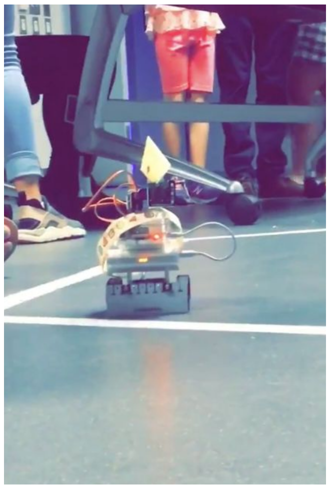
Ultimately, the model was able to recognize and categorize items while incorporating robotic movement. This was tested a number of times prior to the final presentation on Friday June 1, 2018 as seen in the samples we saved from those test runs. However, on the day of our presentation, the url section of our Python code was giving feedback errors in response. None of our previous copies of our codes had worked and the error was with the url. There was an unknown malfunction with the website and our connection to our model and this was an issue for us to access our model altogether. Our code functioned off of a connection to a website to simulate an understanding of the objects and our inability to access our model was a huge issue. This is something that could be improved for further attempts at this method, or possibly just be able to turn to another method to simulate machine learning. The connectivity with our Raspberrypi was waning as well towards the end and perhaps our robot was tired or malfunctioning. More research can be done on this matter and it is valuable to have the knowledge we gained after the complications with our final execution of this project. The final presentation consisted of robotic movement around the area and presentation of some of the pictures it took.



```
pi@robot3:~/Desktop/object-detection-sample-python-master/code $ python picameras
emifinal.py
image has been captured yo
<html>
<head><title>500 Internal Server Error</title></head>
<body bgcolor="white">
<center><h1>500 Internal Server Error</h1></center>
<hr><center>nginx/1.10.3 (Ubuntu)</center>
</body>
</html>

Traceback (most recent call last):
  File "picamerasemifinal.py", line 54, in <module>
    response = json.loads(r.text)
  File "/usr/lib/python2.7/json/__init__.py", line 338, in loads
    return _default_decoder.decode(s)
  File "/usr/lib/python2.7/json/decoder.py", line 366, in decode
    obj, end = self.raw_decode(s, idx=_w(s, 0).end())
  File "/usr/lib/python2.7/json/decoder.py", line 384, in raw_decode
    raise ValueError("No JSON object could be decoded")
ValueError: No JSON object could be decoded
pi@robot3:~/Desktop/object-detection-sample-python-master/code $
```

In the future, we would try to access our training data locally on the pi instead of on the Nanonets website since the API error prevented us from having a presentable project. We would also like to play more with Processing to develop ways of presenting this data in real time. We didn't get as many opportunities to test presentation as we wanted since we dealt with many time consuming connectivity issues. It would additionally be interesting to explore recognition of other objects using deep learning. Overall, we learned a lot from doing this project and gained a lot of experience and knowledge from this course even though our final project didn't work as expected.





- ➢ Link to demo of Robotic Movement: https://youtu.be/j9J9z1f9khA

Distribution of Labor

Research: evenly distributed between Margie and Kara
Arduino Code Modifications: Kara
Processing: Margie
Image Collection: evenly distributed between Margie and Kara
Object Training on Nanonets: Margie
Model Training in Python: Kara
Python Code Modifications: evenly distributed between Margie and Kara
End of the Year Show Presentation: evenly distributed between Margie and Kara

References

I. https://mashable.com/2018/03/07/google-arts-and-culture-ai-experiments/#PJyDIe3Usiql
II. https://medium.com/nanonets/how-to-easily-detect-objects-with-deep-learning-on-raspberrypi-225f29635c74
III. https://www.technologyreview.com/s/611040/facebook-helped-create-an-ai-scavenger-hunt-that-could-lead-to-the-first-useful-home-robots/
IV. https://github.com/RodgerLuo/robotic-vision#ucsb-intelligent-machine-vision-course
V. https://app.nanonets.com/objectdetection/#integrate/107af7b9-ac8a-46ba-8ce7-29ba44ebedf5 (Link to training Data, Model, and API on Margie Fargas's Nanonets account)
VI. Source codes:
    i. http://learningprocessing.com/exercises/chp15/exercise-15-07-image-processing
    ii. https://gist.github.com/sjain07/a30388035c0b39b53841c501f8262ee2
    iii. https://github.com/NanoNets/IndianRoadsObjectDetectionDataset
    iv. https://github.com/RodgerLuo/robotic-vision/blob/master/Autonomous_Walking_Demo/Autonomous_Walking_Demo.ino