Smart Cat Door

Authors: Irene Lin, Philip Petrakian, Jing Wu: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—This system is capable of recognizing cats in a video feed and opening a door automatically. Computer vision detects and tracks movement. Machine learning determines if a cat is present. The goals of the system are to be bidirectional to allow freedom, quiet to not scare the cat, accurate so cats are not locked out and raccoons are not let in, and fast so the cat is not left waiting for the door to open. A mobile app allows the owner to remote lock and unlock the door.**

***Index Terms*—cat door, computer vision, machine learning**

I. INTRODUCTION

The purpose of this system is to keep unwanted animals out of your home while allowing a pet owners cat to have the freedom of entering and exiting. Current smart cat doors use RFID technology and have poor reviews online [5][7][11]. Recurring themes in the product reviews were about the desire to keep raccoons out, have consistent and reliable door opening and locking over extended period of time, and to have longer battery life. Other complaints included the sound of the bolt scaring the cat and the owner having to retrain the cat to push through a heavier door.

In our design, a camera mounted on top of the cat door sends footage to the processor. Computer vision is used to detect and track movement and a machine learning algorithm is used to decide if the animal is your cat. A panel covering the cat door is lifted by a motor and closes after the cat has passed through. Through a mobile app, the pet owner can remote lock and unlock the door.

II. DESIGN REQUIREMENTS

Assumptions:

- The pet will walk towards the camera headfirst
- The pet is not covered in anything that significantly changes its appearance
- A pet owner's house has WiFi

Requirements:

1. Raccoons get in 5% of the time:
    - Reasoning: With current smart and non-smart cat doors, raccoons get in 100% of the time because raccoons are clever enough to lift latches and strong enough to brute force their way in. Any number less than 100% is already an improvement. Facial recognition for humans have around a 5% miss rate, so we decided to challenge ourselves to match this.
    - Results: 20% of the data set was withheld for testing. Please see the Section V Part B: Convolutional Neural Network for more details on test results.
2. Owners cat is stuck outside 5% of the time:
    - Reasoning: If a cat enters and exits four times a day, then across five days, the owner gets one notification. This is reasonable.
    - Results: 20% of the data set was withheld for testing. Please see the Section V Part B: Convolutional Neural Network for more details on test results.
3. 1.1 s for door to open when the cat starts at 1m away from the outside:
    - Reasoning: Cats walk 3.3 km/h (2 mph) on average. When a cat has moved to within 1 meter to the door, we interpret this as the cat having the intention to enter the house.
    $$1.09s = \frac{3600seconds}{1hour} \times \frac{1hour}{3.3km} \times \frac{1km}{1000m} \times 1m$$
    - Results: The motor took 1 second to open the door.
4. Detect when the cat is all the way through and close the door after:
    - Reasoning: The door should not close while the cat is still in the middle of the doorway, as to not injure the cat.
    - Results: The door took 6 seconds to close because the PIR sensors took a long time to settle.
5. Set curfew, remote locking and unlocking, authorize breeds:
    - Reasoning: There are times when an owner does not let their cat outside (e.g. when it is raining outside). The owner needs to be able to set a curfew for when the cat is not allowed outside. The owner also needs to be able to initialize the system to work with his cat's breed type.
    - Results: Although the mobile app could set locking and unlocking, there was no curfew and authorization for breeds implemented.
6. Consistent lighting for nighttime:
    - Reasoning: Computer vision requires consistent lighting and the camera needs to be able to obtain clear footage of the cat even at night.
    - Results: Although the LED was mounted on the door, no testing was done.

7. Sturdy enough to withstand 13.6kg of resistance:

    - Reasoning: A research study [9] conducting strength testing on raccoons indicated that adult raccoons can lift 13.6kg of resistance.
    - Results: Using a hanging digital luggage scale, we measured the door to resist at least 9kg of force.

8. Operate under 70dB:

    - Reasoning: This number comes from the sound intensity of a vacuum cleaner. Cats are often afraid of fireworks, thunderstorms, and vacuum cleaners, with vacuums being the most quiet of the three [4].
    - Results: Using a decibel reading app, we measured the loudest part of the door's operation to be the clicking of the solenoid. It peaked at 40dB from 1 meter away.

Full System Testing: Due to limitations on time, full system testing was not conducted.

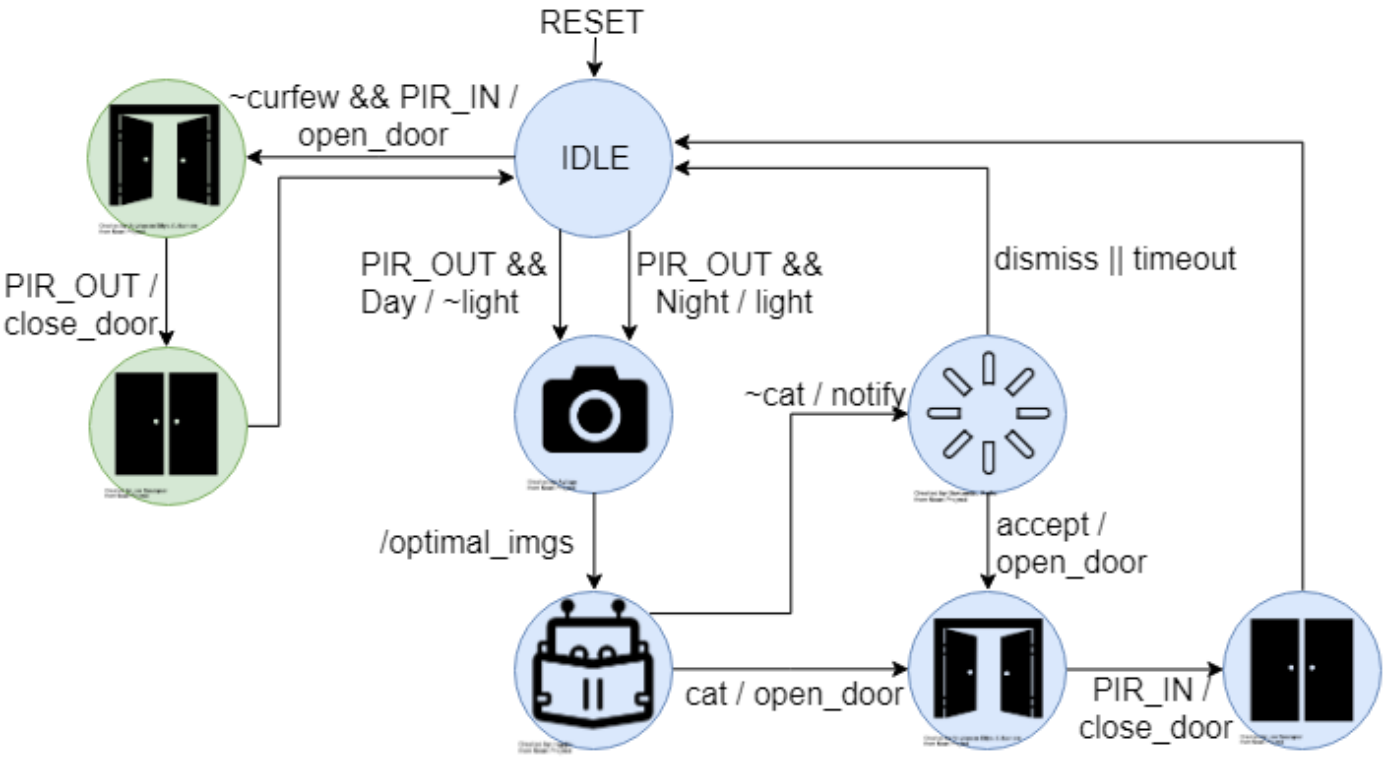

Figure 1: Event State Diagram

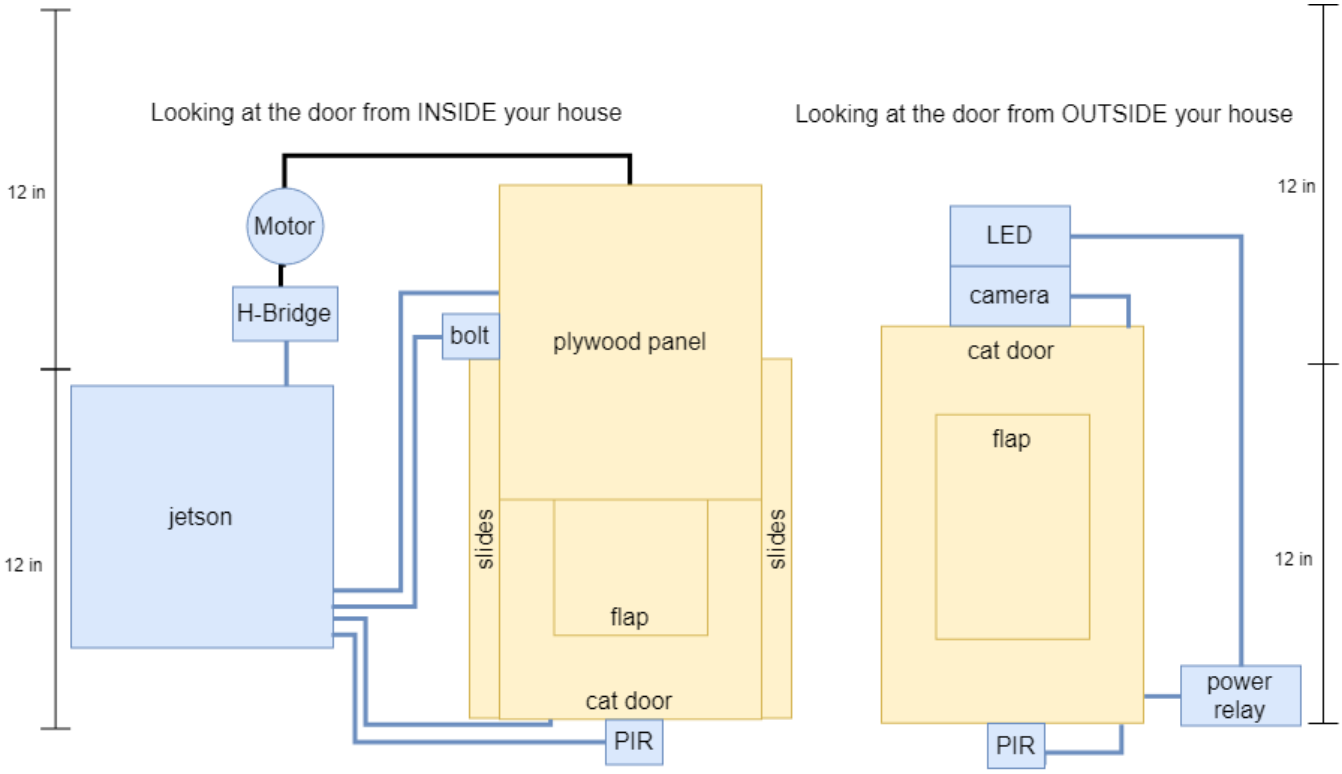

Figure 2: Door Schematic

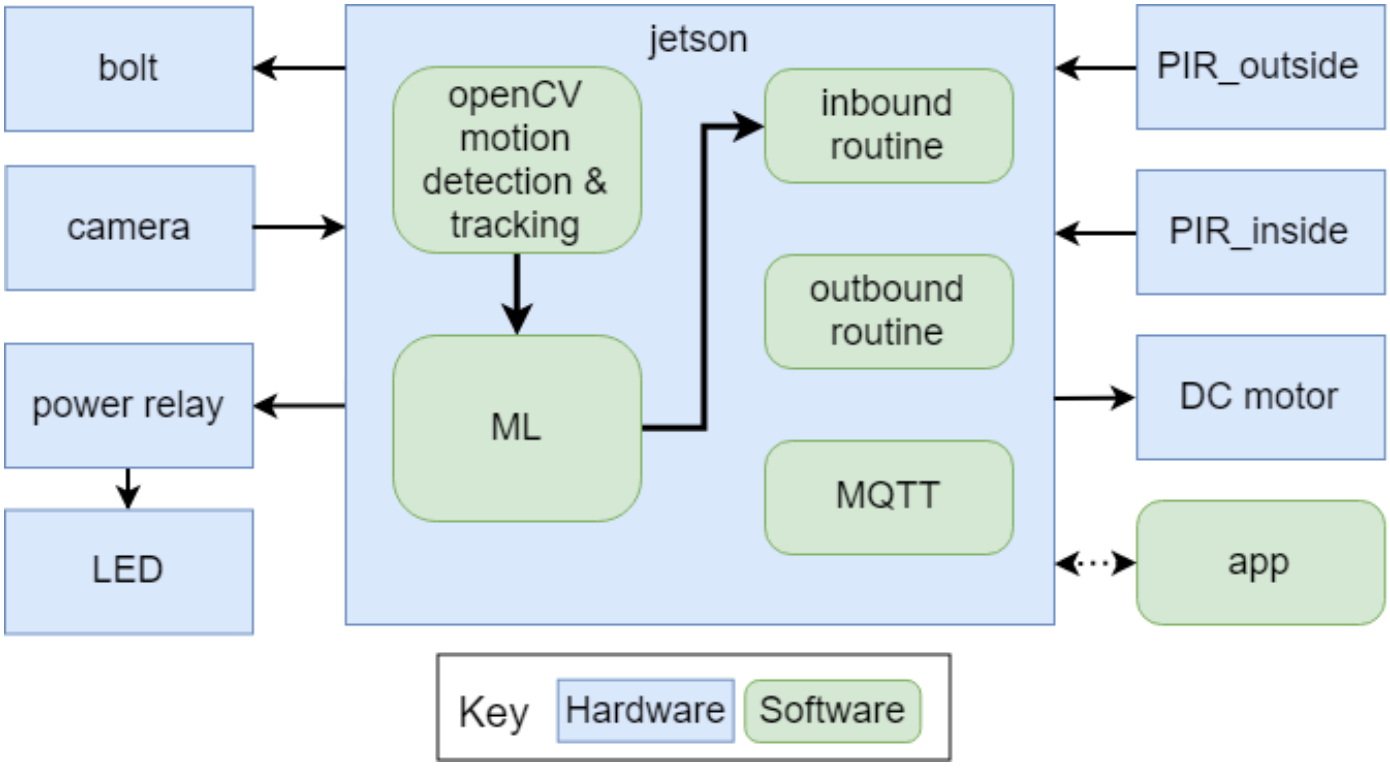

Figure 3: Block Diagram

III. ARCHITECTURE OVERVIEW

Figure 1 illustrates a state diagram of the system. Blue states are for entering, green states are for exiting. For a cat wanting to exit, the PIR sensor mounted on the indoor side of the cat door detects when the cat has moved close to the door. If it is not curfew, then the door opens. After the cat has passed out of range of the outdoor sensor, the door closes. For a cat wanting to enter, the camera detects and tracks movement, and PIR sensor on the outdoor side of the cat door detect when a cat has moved close to the door. If it is night time, the light turns on. This helps the camera to produce better images to feed into the machine learning algorithm. The machine learning algorithm determines whether or not the door should be opened. In the case that the algorithm decides the door opens, the door does not close until both sides see no motion on the PIR sensors.

Figure 2 depicts how the devices and non electrical parts are linked. Blue indicated device and yellow indicates non electrical component. Blue lines indicate wires. The diagram is drawn to scale.

Figure 3 describes how software and hardware components interact. Blue is hardware and green is software. Motion detection and tracking is implemented in Python and openCV. The machine learning algorithm is implemented using Python and TensorFlow. The inbound and outbound routines handle the PIR sensors, power relay, solenoid bolt, and servo. The mobile app communicates with the Jetson through the MQTT protocol and is implemented using Xcode in Swift.

IV. DESIGN TRADE STUDIES

A. Motion Detection

Motion detection is an important part of our project because we need to know when an object has passed in front of the door before we can begin to classify the object. While a PIR sensor alone can detect motion within a certain proximity, our primary method of detecting motion uses a motion detection algorithm on our video feed. The PIR sensor acts as a secondary sensor which turns on the LED lights at night when something walks in front of it, providing the camera with sufficient lighting to send a visible video feed to the Jetson board.

We chose to use video motion detection over a traditional proximity sensor because video motion detection provides better images to our machine learning classifier, in addition to detecting motion. When taking pictures of small animals from one to two meters away, the animal may take up less than $\frac{1}{9}$ of the entire image space using a traditional phone camera. By being able to detect where objects are in the image, we can crop that part of the image out and only send the cropped image to our machine learning classifier, resulting in a more accurate and consistent output.

B. Machine Learning Classification

Our project ultimately aims to do facial recognition by examining the specific features of cats (eyes, ears, nose, fur, etc.); however, due to a lack of available data sets, the time frame for the project, and the skill set that we possess, we decided that it was unfeasible to label features of the cats and train on those features in the time frame of the project. Instead, for the first iteration of the cat door, we are identifying cats and allow them to enter and exit.

Convolutional neural networks are the most popular and successful image classifiers today and acts as our machine learning classifier [3]. Most convolutional neural networks use at least one of each of the following layers: convolution, activation, max pooling, fully connected (dense), and softmax inference. We decided that having one of each layer would suffice because the sample space of possible images is small - our camera is stationary and is focused on an area close to the ground. Therefore, classification naturally is more accurate and does not require a complicated neural network. Animal recognition research done at the University of Zilina [12] showed that a convolutional neural network with one more convolution and activation layer than our neural network successfully recognized animals on average 95% of the time. So our original goal was set for 95% accuracy

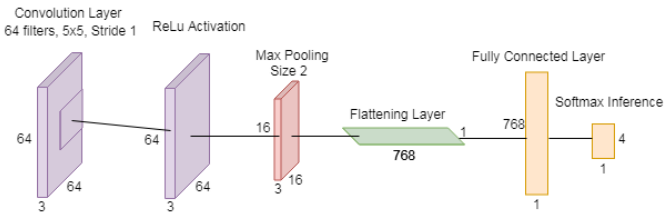

Figure 4: Initial Convolutional Neural Network

C. Testing Tradeoffs

In order of preference, the testing options are as follows: live animals, taxidermy, video feeds, printed high resolution pictures, and stuffed animals. Irene's friend has a cat and Jings friend has a cat, but animals arent allowed on campus. The best feasible testing method would have been to record footage of said cats interacting with the system and the system responding appropriately. For raccoons, we were not able to find live raccoons to test our system on. A taxidermy raccoon is as close as we could get to a live raccoon, but they are expensive. We looked for videos of raccoons facing headfirst at the camera such that it is very similar to the video feed the camera would have captured of a raccoon. Stuffed animals are not desireable as a test subject because they do not look like real animals. A machine learning classifier that classifies a cat plushies as a real plushie is a poorly built classifier. We were better off printing high resolution photos of cats and raccoons. Unfortunatly high resolution photos provided a set of images of cats that were sitting on couches or in grass. They were not entirely representative of what a cat would have looked like walking up to a cat door.

D. Hardware Tradeoffs

We wanted to minimize the latency of our computer vision and ML algorithms because we want to be able to open the door for a valid cat as it is walking up to the door, without having the cat needing to wait. We estimated that the cat will be within range of the camera for a total of 1.2 seconds.

Through our research we determined that a Raspberry Pi would allow us to compute around 1 frame per second, which is too slow because we could potentially only receive one image during the 1.2 second span and this image might not give a good indication of whether the animal is valid or not. Similarly, we looked into Odroid which is a board similar to the Raspberry Pi, but much more powerful[2]. This would likely yield us 2-3 frames per second. Still, we are unsure if this frame rate is fast enough and we want to be sure that we are going to get at least one good image for our algorithms.

We then looked into GPUs, which are processing units designed for image processing. Nvidia makes the most commonly-used and best documented GPUs. In addition, one of our group members has experience with Nvidia GPUs. We found the Jetson family, which are GPUs created for the embedded systems world. Specifically, we chose the Jetson TX2, which has 256 Cuda cores, because based off of our research we would be able to process 15 frames per second. Furthermore, Nvidia has a library called TensorRT, which compliments TensorFlow. This library can be used in conjunction with TensorFlow to optimize the ML algorithm computation for Nvidia GPUs. Although we originally planned on using TensorRT to speed up our system, we discovered after testing that we did not need it, as we already could get over 15 frames per second.

Power is not listed as a requirement or metric because we will never be making a decision based on power. The device is plugged into a wall outlet.

E. Door Construction Tradeoffs

The two options for pet doors are a swinging door and a lifting door. Online reviews for swinging doors indicated that raccoons were smart enough to lift latches and strong enough to brute force their way through cat doors. We decided to go with a lifting door using drawer slides oriented vertically. This is much harder for a raccoon to muscle through. A solenoid bolt is used to prevent a raccoon from sliding the door upwards.

V. SYSTEM DESCRIPTION

A. Motion Detection

The motion detection algorithm is implemented in Python using the OpenCV library. First, we store a weighted average of previous frames and call this our background frame. With the weighted average, the script dynamically adjusts to the background, even as the time of day changes along with the lighting conditions. The current frame is smoothed and converted into grayscale, and then the absolute difference is taken between this and the reference frame. A binary threshold is applied to the frame delta such that if a pixel's intensity is above a set threshold, the pixel is set to maximum intensity. For each contour in the frame delta, if the contour area is greater than the minimum contour area parameter, the contour is considered to be actual motion. Suppose the wind blows and the grass bends. This movement is not considered motion because the area of change for a rustling blade of grass is too small. The image is cropped around all significant contours. In order to track where to crop, a bounding rectangle is drawn around each contours and the outermost corners out of all the contours set the cropping region. The cropped frame is passed to the Convolutional Neural Network and the smoothed grayscale current frame is saved as the new reference frame. Because the motion detection saves a new reference frame as soon as the delta is large enough, it does not matter how slowly something moves towards the door.

B. Convolution Neural Network

Once motion has been detected, the image stream is fed into our Convolutional Neural Network for classification. For our inference, we classify between 4 number of entities, as our cat door would be stationary in front of a door and the number of different objects that could come into the camera view is few. Adding more entities is trivial, as we can download a set of images of the new entity from an online database and retrain our model. The current list is as follows:

- Cat
- Dog
- Raccoon
- Lower Human Body

Initially, we had planned to not only classify between the above 4 classes, but also classify between breeds as a first step to doing "face recognition" for cats. However, we could not get enough labeled training data for this, and were only able to achieve a 40% validation accuracy with a dataset of 2400 images of 12 different breeds of cats.

The data set that we began with had 5200 images: 300 images of raccoons, 500 images of humans, 2200 images of cats, and 2200 images of dogs. The images of the animals were from all angles, from the front-view, side-view, and back-view. However, almost all of them contained some view of the head or face. The images of the lower human body were images of humans from the hip down. Additionally, we flipped each image across the vertical axis so our data set size was doubled. Images were also rotated between -15 and 15 degrees at random to make our system more robust to noise.

Our initial architecture (see Figure 4), using this data set, achieved a validation accuracy of 72%. We then experimented with various architectures by changing the number and sizes of layers. We also tweaked parameters including the activation function, learning rate, batch size, and we were able to improve the accuracy to 76%. One of the main bottlenecks for our neural network was that our data set was too small so we found more images and increased the size of our data set to 8000 images. After training the convolutional neural network over 50 times, we achieved a validation accuracy of 84%.

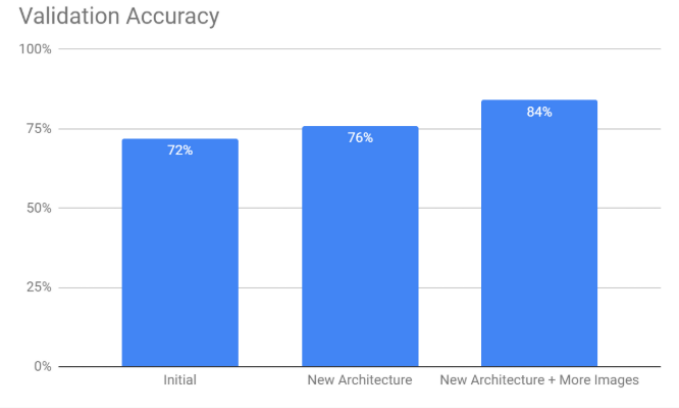

Figure 5: Validation Accuracy over Iterations of Training

This validation accuracy was the result of our training, but it does not show how robust our system is under real-world conditions. Real world conditions include factors such as poor lighting and unclear images which may confuse our system. When live testing our convolutional neural network on a live video feed, we sometimes fed images into our machine learning classifier which did not contain the entire object, contained an object which was too zoomed in, or contained an image that was the wrong size. For example, a close up image of a human leg may be in a rectangular shape. When that rectangular image is fed into our neural network, it is resized to a square shape and the classification is distorted. Because our neural network was not trained on images which were noisy, unclear, and of the wrong dimensions, when testing on printed images of animals, we discovered that edge cases caused our classification to be incorrect. One common edge case is when an object first enters the view from the side. Our computer vision algorithm begins feeding images to the machine learning classifier before the object is completely in the view, thereby only looking at part of the object and identifying it incorrectly.

In order to compensate for this, we decided to classify the most recently seen 10 frames, and if 8 of the 10 frames are cats with 80% confidence, then we classify that object as a cat. Right now the system is too

kind. It lets in too many things. But compared to the reverse problem of being strict on what to let in, our problem is easier to solve because we can add more restriction layers by increasing the number of frames and the confidence interval that is required to classify an object. Otherwise, we do not see any object and consider it as "None."

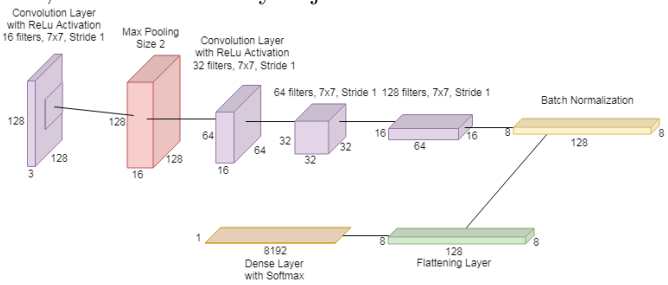

Figure 6: Final Convolutional Neural Network

C. Automatic Door

The door was constructed using two squares of plywood. A rectangle was cut in both plywood panels and the cat door was installed in the hole. A 21cm x 23cm plywood panel was laser cut and secured to the drawer slides. Professor Nace helped with the band saw to create two rectangular prisms for the other side of the drawer slides[6]. We secured the drawer slides to the rectangular prisms and attached them to the indoor side of the door. We also mounted a DC motor directly above the sliding door panel. Using a fishing wire, the motor was able to generate enough torque and power to open the door in 1.6 seconds and close the door in 1 second.

On both sides of the door, two small holes were drilled 2.5cm apart below the cat door hole. Screws were placed in these holes in order to mount the PIR sensors. The camera is mounted on the top of the outdoor side and angled downwards. The LED is mounted above the camera. The solenoid bolt is mounted at the top of the right rectangular prism on the indoor side. It is positioned such that when activated, it blocks the door from opening, and when not activated, it does not obstruct the functionality of the door. Testing revealed that the bolt is the noisiest part of the system. Using the DecibelMeter app, we measured the solenoid's click to peak at $\tilde{4}$0dB from 1 meter away when activated[13]. The strength of the solenoid was tested using a hanging digital luggage scale[1]. The door can sustain against at least 9kg of upwards force.

Passive infrared sensors detect changes in infrared radiation. All objects with a temperature above absolute zero emit heat energy in the form of radiation, so a PIR sensor can be used to sense movement of people, animals, or other objects. Although the door was able to open and close quickly, the PIR sensor took time to settle, therefore it took 6 seconds for the system to decide to close the door, and one second for the motor to rotate until the door was closed. This is an issue for a pet owner because another animal could easily tailgate through the door. In order to improve the system, we would need to utilize a sensor that reacted to the environment more quickly than this PIR sensor.

The LED Photography Lighting Kit is commonly used for photography studio, lighting for video, images, collocation with all kinds of tabletop studio, and video shooting. It provides consistent lighting for the camera.

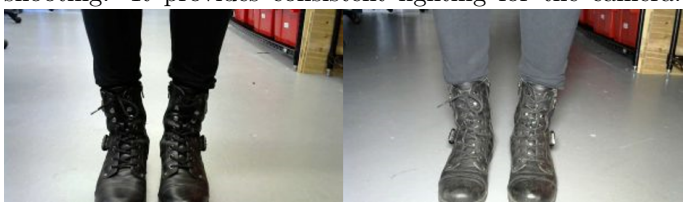

Figure 7: (Left) Camera output without the LED. (Right) Camera output with the LED.

D. iPhone Application

We wanted the user to be able to communicate to the smart pet door at any time, not necessarily within proximity of the door. Bluetooth communication is done within close proximity within devices, whereas Wifi works as long as both devices have a connection. Therefore, the smart pet door is connected to Wifi through the Jetson development board. We are assuming that the whole house has a Wifi connection. The Jetson is able to receive commands from the user and send data back to the user. We used the MQTT protocol to communicate between the two devices as the cocoaMQTT library can be implemented in Swift (for the iPhone app) and in Python (for the Jetson). This is a pub/sub form of communication which is easy to use if a user has more than one pet door.

In our app, the user is able to lock and unlock the door being activated in the outbound direction. With this feature, pet owners are able to prevent their cats from leaving the house. In addition, the app indicates the most recent door use, including the direction and time of use. The time for the door to lock or unlock occurs nearly instantaneously. However, because this system requires Wifi, a poor Wifi connection would increase the update time.

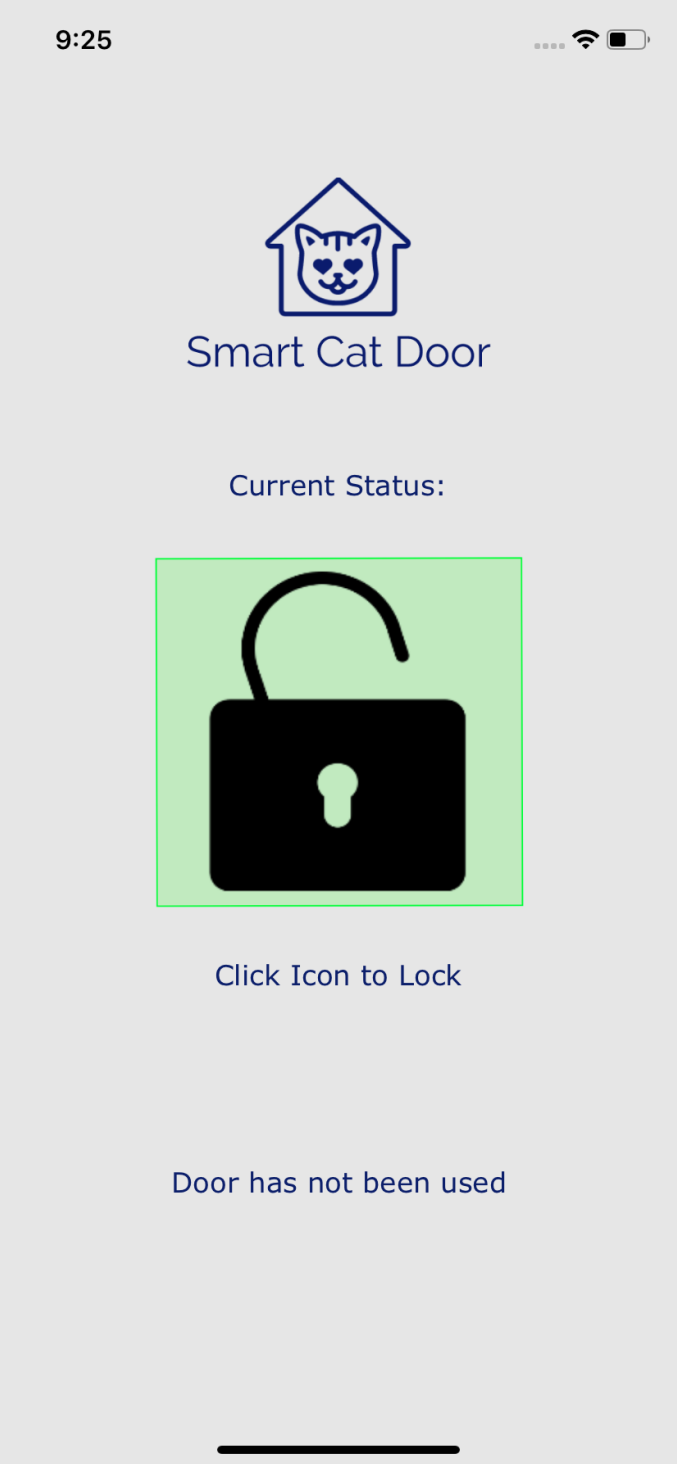

Figure 8: Screenshot from iPhone app

E. System Hub

Our system is able to communicate over Wifi to a phone, receive camera footage, apply our Computer Vision and ML algorithms, control the servos for the door, turn on and off and LED, and receive PIR data. Originally, our plan was to use a Raspberry Pi to accomplish all of these requirements. However, after selecting the Jetson TX2 to compute the Computer Vision and ML algorithms, we discovered the Jetson TX2 developer kit. In addition to having the Jetson TX2 GPU, this board also has a quad-core Arm based processor, 8 GB of memory, 32 GB of flash storage, USB ports, GPIOs, and Wifi capabilities. It also runs a Linux OS. In fact, this developer kit has the same set of features as a Raspberry Pi, that we would need to use for this project. Therefore, we concluded that the Raspberry Pi would add unnecessary complexity to our system. As previously discussed, we are using the MQTT protocol for Wifi communication. We wanted a camera that would be able to compute at least 15 fps as we expect our algorithms to be able to process images at 15 fps. In addition, the video quality we require is approximately 100 pixels by 100 pixels. Modern cameras easily satisfy both those requirements, we decided to go with a 720p, 30 fps camera that uses USB to communication with the developer kit. Next, the motor for the door, the LED, and the PIR sensors can communicate using with the developer kit over the GPIO pins. We wanted to be able to detect movement within a 2 meter radius of the door as this is when we wish to activate the opening or closing of the doors. We chose the PIR sensor because it uses heat signatures within a specified radius to detect how far movement is. It outputs the values using a single wire.

Our final system, is a Python script that controls and uses all aspects of our system. It receives inputs from both PIR sensors, the ML algorithm, and from the iPhone app. It outputs to open and close the door and sends data to the iPhone app. The script also includes the decision algorithm for inbound movement. It keeps track of the past ten frames outputs. If eight out of the ten previous frames are predicting a cat with 80% confidence, then the door is opened. In the outbound direction, the door is opened if the PIR sensor is ever activated, as long as the user has not locked the door through the iPhone app.

VI. PROJECT MANAGEMENT

A. Schedule

We encountered various setbacks along the way, which put us behind in the schedule. If you look at our schedule, because of debugging various compatibility issues and hardware components, we fell behind a lot. See Gantt chart in Appendix A.

B. Team Member Responsibilities

Each team member is primarily responsible for technical portions of the project, and secondarily responsible for the integration portions and the management portions of the project. Irene is primarily responsible for constructing the door and detecting motion. Philip is primarily responsible for implementing the mobile app, integrating all of the software components within the Jetson (sensor triggers and algorithms), and accelerating the ML computation on the GPU. Jing is primarily responsible for training the machine learning classifier and integrating it with the Jetson GPU.

Irene completed door construction, motion detection,

and some testing. Jing completed the machine learning model. Phil completed the mobile app, Jetson bring up, and PIR sensor control. Irene and Jing completed solenoid circuitry and wiring. Phil and Jing completed the motor and Jetson integration. Irene and Phil completed the system control flow.

C. Budget

In addition to the parts listed in the table below, we also used $300 worth of AWS credits for training our model. Please see the AWS Credit Usage section at the end of the paper.

Part Name	Quantity	Price	Not Used	Realized we needed
Logitech C270 Camera	1	$19.86		
PIR (motion) sensor	2	$19.90		
Cat door: 5.125in x 7.5in	1	$29.95		
High Torque - Metal Gear Servo	1	$19.95	x	
Machined Aluminum Servo Arm	1	$4.50	x	
Bottom Mount Drawer Slides	1	$5.28		
Plywood	4	$23.45		
LED	1	$34.99		
Power Relay	1	$19.85		
Solenoid	1	$18.69		
BS170 MOSFET	1	18-220 lab		x
Jetson TX2	1	Paid in full		
12V Battery	1	$17.99		x
L298N Motor Driver	1	$6.99	x	
Continuous Rotation Servo	1	$11.95	x	
Motor Driver 2A Dual L298 H-Bridge	1	$19.99		x
TB6600 Stepper Motor Driver	1	$34.95	x	
NEMA 17 Stepper Motor	1	$14.00	x	
DC Motor	1	Build18	x	
18-349 Motor	1	18-349		x

Figure 9: Table of parts

D. Risk Management

In the design phase, anticipated risks included not having enough time to test, integration of peripheral devices with the Jetson, and an incomplete machine classifier. The risks we actually faced along with their respective mitigation techniques were as follows:

1. It turns out the Jetson's GPIO can only output 3.3V, but our transistor plus solenoid requires 4V to turn on, so we cannot turn on the solenoid with the circuit as is. However, we do have a 5V power source, so we came up with another solution to turn the solenoid on and off. We used a second transistor that can turn on at 3.3V to connect and disconnect the 5V power source to the original transistor. The preventative measure we should have employed during the design phase was to research the components and read the specifications more carefully. We would have found that the solenoid has a 1-10 seconds long activation time, which is an issue given that the door would be locked for much longer than that. We would have also found that the particular transistor we were using with the solenoid has 5ohms RDS, which is pretty sad relative to other transistors in 2019.

2. We met obstacles in getting a motor to work, so we set up multiple meetings with TAs in order to figure out solutions. We also ordered parts for multiple possible solutions and tested each. This helped our team in identifying why particular off the shelf components did not work, but unfortunately did not help in engineering a working motor solution to use in our project and caused delays in the integration phase of our schedule. We surveyed multiple motors before getting a DC motor to work. Unfortunately, it rotated at about 60rpm, which is much too slow for our system. The next motor we found operated at 200rpm. This combined with fishing line solved our problem.

3. When we were running behind on the schedule, we cut away at unnecessary improvements such as optimizing the ML algorithm for the Jetson. The Jetson board was powerful enough and the machine vision simple enough to where these improvements were considered unnecessary. Even without TensorRT, our machine vision script operated at 16fps on the Jetson. Cutting away at these stretch goals was the right decision.

4. Initially, with only 300 images of raccoons and 500 images of humans, our data set was too small. To make our system more robust to noise and to enlarge the number of samples we had, we rotated each image across the vertical axis to double the size of our data set. Additionally, while training, each image was rotated -15 to 15 degrees randomly to make our neural network responsive to images which are not directly straight.

5. When inference reached a peak of 76% accuracy on the validation data set, we had only two choices to modify our neural network our increase the number of images we had. Tweaking the neural network to improve accuracy took too much time - approximately 4 to 6 hours per iteration. Therefore, we decided that the simplest way to improve accuracy without spending too much time is to enlarge our data set. We scraped Google for more images until we doubled the number of raccoon and human images we had, because those were the ones we were lacking the most.

6. For the demo, we were not able to set up with both sides of the door protected against random passerby movement, so we disconnected the PIR sensor on the indoor side for demo purposes. This solved our problem for demo purposes, but it really only swept a bigger problem under the rug. At the root of the problem, the PIR took too long to settle, so the system took seconds in order to decide to close.

VII. RELATED WORK

A. Regular cat door

The obvious original design is a simple cat door with a flap, that a cat or any animal is able to go through. The clear problem is that any animal can get through the door, and can potentially wreak havoc within the house. In addition, some cats never gain the knowledge or confidence to use a cat door because it is not intuitive for them.

B. RFID activated cat door

An alternate to the cat door is an RFID activated cat door. This system works by placing a collar on the cat with an embedded RFID tag. The benefit to this system is that it only allows your cat in, assuming it is not next to the door with another unwanted animal nearby. The downside with this system is that a lot of cats easily lose their collars or simply refuse to wear one. In addition, as previously stated, some cats do not like using a door with a flap.

VIII. SUMMARY

While our final product doesnt meet our initial expectations of having a 5% chance of a false negative, or a 5% chance of a false positive, it is still a great improvement from current commercial products. First, it keeps out raccoons more than 90% of the time. Although we achieved a validation accuracy of 84%, raccoons have many more consistent and distinct features from cats, dogs, and humans. When testing on only our raccoon data set, we discovered that raccoons were classified as raccoons 90% of the time. Therefore, raccoon entrances into the house should be unlikely. Whats more is that the user has the option to lock the door in the outbound direction, and can can see the most recently used action with the door from the iPhone App. This is great improvement from current cat doors, which cannot be locked or keep track of entrances. Furthermore, The automatic door is also high tech and improves the cats standard of living.

A. Future Work

i. Self Learning Algorithm: If we were to do this project again, we would design a self learning algorithm. For the first few times a cat walked up to a door, the camera would take a picture and the processor would notify the owner. The owner would see the picture and have the option to open the door remotely. Over time, the algorithm would learn who should and shouldnt be let in, notifying the owner less and less.

ii. IR Lighting: An alternative lighting mechanism would be IR lighting. IR wavelengths of 850nm and 940nm (also called NIR Near InfraRed) are commonly used in machine vision. IR reduces color of objects, glare, and reflections. IR has a longer wavelength than visible light which usually results in a greater transmission of light into a material through materials like paper, cloth and plastic. IR wavelengths react differently on materials and coatings than visible light, so certain defects and flaw detection can be identified with IR where visible light did not work. One drawback would be that IR lighting changes the color of the cats fur in the image and therefore, our machine learning model would have to be trained on IR images. This dataset is hard to find.

iii. Zero Weight Counterbalance Spring: The door does not have a safety mechanism for opening and closing. In order to protect cats passing through the door, a zero weight counterbalance spring would be used with the doors lifting panel and servo. This would allow the door to open faster, but close slowly. The door would spring upwards rather than down on the cat in the event of a malfunction.

iv. Audio Processing of Cat Meows: We set a target false positive of raccoons let in 5% of the time, because this is already a great improvement over current cat doors, both smart and non-smart. But this means that if a raccoon attempts to enter a house through the cat door once a day, then a raccoon would succeed once every three weeks. The machine learning model could be used in conjunction with audio processing of cat meows in order to increase the accuracy and decrease the percentage of false positives. Cats have complex vocal chords, and therefore different cats have different purrs and meows[10]. A project has been done to build a classifier for meows and woofs[8]. Future work would build on this research.

B. Lessons Learned

1. When working with Machine Learning, check the availability of desired datasets. We did not verify that we would have enough data, so we assumed an adequate amount of data was available and over promised during the design phase.

2. We realized too late that we should have recorded footage of a cat simply walking around at all angles and directions through the same camera as the one used in our system and mounted at the same height and angle. This would've allowed us to have a test vector that we can feed into the motion detection algorithm and machine learning. We would've observed the inference results and it would have allowed us to make better design decisions regarding the control flow of the system. The footage would've been exactly what the camera would have saw.

3. Start on hardest and most unfamiliar aspects first. Start everything at least a little bit in order to better gauge how difficult it will be, and to adjust schedule and resources accordingly.

4. Do as much research as possible. A lot of the problems we encountered could've been avoided if we had read documentation more carefully and researched more thoroughly.

5. Ask for help on getting help. Being more proactive and asking your friends or colleagues for help was something that we did benefit from and could have done a lot more of.

C. AWS Credit Usage

In order to do train our neural network, we needed a computer dedicated for training. One choice we had was to use AFS, the Carnegie Mellon University computing clusters. However, due to university restrictions, we likely could not install the libraries required for machine learning on those machines. Instead, we decided to go with Amazon Web Services, which has dedicated servers specifically for machine learning.

In total, we received $250 of AWS credits, but went about $100 over our budget, paid out of pocket, totaling to approximately $350 spent on AWS. AWS was primarily used to train our convolutional neural network. We spent one month fine-tuning our neural network on AWS, training it over ten times a week to see improvements. The AWS instance that we chose was a p2x.large, an instance specifically designed for Machine Learning. Because we were not familiar with convolutional neural networks, many training iterations did not produce better results, but instead were a test to see how a change in certain parameters and functions affected the output of the network. The training took approximately 2-6 hours depending on the parameters that we set and the data set that we were training on, so most of the time, we let it run once over night and once during the day.

We are extremely grateful for the AWS credits as our Machine Learning could not have been possible without them. Training the convolutional neural network would have taken 40 hours on our personal computers, but with AWS, we were able to do it in less than 6 hours.

References

[1] Amazon. *Etekcity Digital Hanging Luggage Scale*. URL: https://www.amazon.com/gp/product/B00NW62PCA/.

[2] LTD. HARDKERNEL CO. URL: https://www.hardkernel.com/.

[3] Sustkever Krizhevsky and Hinton. *ImageNet Classification with Deep Convolutional Neural Networks*. Jan. 1, 2012. URL: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[4] Litter-Robot. *Why Are Cats and Dogs Scared of Fireworks?* URL: https://www.litter-robot.com/blog/2018/06/29/cats-and-dogs-scared-of-fireworks/.

[5] Cat Mate. *Elite I.D. Disc Cat Flap with Timer Control*. URL: https://www.amazon.com/Cat-Mate-Elite-Timer-Control/dp/B000XPSH34.

[6] Bill Nace. URL: https://www.ece.cmu.edu/directory/bios/nace-bill.html.

[7] PetSafe. *Electronic SmartDoor, Automatic Dog and Cat Door*. URL: https://www.amazon.com/PetSafe-Electronic-SmartDoor-Automatic-Activated/dp/B000WJ0IGA.

[8] Summer K. Rankin. *Classification Of Meows And Woofs*. Nov. 7, 2017. URL: https://www.summerrankin.com/dogandponyshow/2017/10/16/catdog.

[9] Nathan P. Snow. "Strength Testing of Raccoons and Invasive Wild Pigs for a Species-Specific Bait Station". In: *Wildlife Society Bulletin* 41.2 (2017). DOI: https://www.aphis.usda.gov/wildlife_damage/nwrc/publications/17pubs/REP\%202017-031.pdf.

[10] Melinda Story. *Kitty Sounds Cat Noises and their Meanings*. July 14, 2017. URL: https://www.thehappycatsite.com/cat-noises/.

[11] SureFlap. *Microchip Cat Flap*. URL: https://www.amazon.com/SureFlap-Microchip-Cat-Door-White/dp/B003EGIM3O.

[12] Trnovszky. *Animal Recognition System Based on Convolutional Neural Network*. Sept. 1, 2017. URL: https://www.researchgate.net/publication/320162958_Animal_Recognition_System_Based_on_Convolutional_Neural_Network.

[13] Wang Yuan. *DecibelMeter*. URL: https://itunes.apple.com/us/app/decibelmeter-measure-db-level/id1274038012.

Appendix A

	sat feb 16, 11am		mon feb 18, 10:30am		wed feb 20, 10:30am		sat feb 23, 11am	mon feb 25, 10:30am	wed feb 27, 10:30am	sat mar 2, 11am	mon mar 4, 10:30a	wed mar 6, 10:30a
irene	redraw door design & paragraph in the design doc		compile and finalize parts list and submit		refine and seek feedback on block diagram. have finalized block diagram in design doc		work on design presentation	**irene has comp arch test**				
	decide how lighting will be triggered and controlled, decide what mic to use and how audio will be processed.		~~include audio and lighting in design doc~~				lighting, redo diagrams				Design Document	Ethics Document
			add cost requirement to design doc				block diagram					
phil	research communication between app and rpi, include in design doc		research time it takes a cat to walk through a door and decide on a latency requirement, include in design doc		**ML test (Thursday)**		Write up doc for Jetson and its application with TensorRT		**Parallel Test**	Research NVIDIA jetson digits	Design Doc	Ethics Doc
	come up with list of pros and cons for using an Nvidia GPU				Research applicability of TensorRT		Finish doc for app (track entries and exits, publish cat breed to jetson)			Resarch how to convert TensorFlow to TensorRT		
	research/ask friends how to best structure the steps to cv and specify in design doc with reasoning for each step.		email one expert friend / ta / anyone and get feedback from one expert on the ml plan. revise if needed		research how often raccoons enter homes		Find a camera, explain why it fulfills our needs in the design doc			Team status report		
jing	research/ask friends how to best structure the steps to cv and specify in design doc with reasoning for each step.		email one expert friend / ta / anyone and get feedback from one expert on the ml plan. revise if needed		have steps written down of how the each step of the cv/ml algo will be implemented. ie aws for training, X dataset for testing		Dataset, CNN design doc\ why?		Determine layers and parameters of CNN	Build and code layers	Design Document	Ethics Document
	research success rate of other animal recognition algos --> pick percentage for metric --> state reasoning in requirements section of design doc.		Research hardware requirements for tensorflow + openCV on RPi		Determine goals of false positives and false negatives		Flow chart / blcok diagram of ML algorithm			Download training data set		Train model on AWS
	research what cv algos and setups for animal recog have been done and summarize in design doc (include references), make a list of relevant related research papers for irene and phil to read.				Flow chart / blcok diagram of ML algorithm		cat reddit poll					
team	new schedule, team status update		first draft, parts list				final draft completed, ppt completed	design rev			Design doc due	ethics doc due

	sat mar 9, 11am	mon mar 11, 10:30 am	wed mar 13, 10:30am	sat mar 16, 11am	mon mar 18, 10:30am	wed mar 20, 10:30am	sat mar 23, 11am	mon mar 25, 10:30am	wed mar 27, 10:30am	sat mar 30, 11am	mon apr 1, 10:30am	wed apr 3, 10:30am
irene	spring break						have assembled door parts		have servo working with door	have motion detection script v1 done	**Demo in Lab**	**Demo in Lab**
							have video feeds of raccoon			have video feeds of cat		
				team status report								
phil	spring break							Run a python script and pip install libraries on Jetson		have pir set up	have camera working with OpenCv on Jetson	**ML exam (Thursday)**
	spring break										have GPIO demo with input/output	gone for researc trip
							team status report				have clean demo for iPhone app	
jing	Spring Break				upload dataset to aws		have first prelim testing done		have inference imported to jetson		**Demo in Lab**	**Demo in Lab**
	Configure model and retrain over spring break			Use Testing data set on AWS							Find more images and retrain	Test solenoid an check output voltage <13V
	team status report									team status report	Integrate with CV	Order battery
team											have indv parts done, spend april integrating	

	mon apr 8, 10:30am	wed apr 10, 10:30am	sat apr 13, 11:00am	mon apr 15, 10:30am	wed apr 17, 10:30am	sat apr 20, 11am	mon apr 22, 10:30am	wed apr 24, 10:30am	sat apr 27, 11am	mon apr 29, 10:30am	wed may 1, 10:30am	sat may 4, 11am
irene	comp arch midterm		test motor controllers with nema 17		test dc motor	test solenoid sound	start poster design	**Demo in Lab**	test solenoid strength	**class presentations**		update project paper
	order alternative stepper and motor parts									poster due tuesday 4p		
						team status report						
phil		get .hrf file running on jetson w/tensorrt		**Parallel exam**	integrate mqtt and gpio with tensorflow on Jetson		clean up app	**Demo in Lab**		**class presentations**		Test entire system
							finish testing			poster due tuesday 4p		Shoot demo video
			team status report									
jing	Add "None" prediction		Retrain with new images	Debug solenoid circuit		Out of town		**Demo in Lab**	Prepare for presentation	**class presentations**	Debug solenoid circuit and motor	Test entire system
	Retrain with flipped, rotated images									poster due tuesday 4p		print images for demo
							start presentation		team status report			
team									slack	whatever else needs to be done	just in case	the final sprint

Figure 10: Gantt Chart