

Team Fox Wire
AI DASH CAM FOR AMBER ALERTS

Team members:

Sean Atangan

Eric Escobedo

Sai Arkar Khant

Anh Minh Ngo

Table of Contents

<u>Page # 4</u>	<u>Abstract</u>
<u>Page # 5 - 11</u>	<u>Project Selection and Scoping</u>
1. <u>Projects Goals</u>	
2. <u>Needs Statement</u>	
3. <u>Objective Statement</u>	
4. <u>Background</u>	
5. <u>Marketing Requirements</u>	
6. <u>Objective Tree</u>	
<u>Page # 12 - 15</u>	<u>Engineering Requirements</u>
1. <u>Engineering Requirements List</u>	
2. <u>Marketing Requirements Table</u>	
3. <u>Marketing Requirements List</u>	
<u>Page #16 - 20</u>	<u>Design Alternatives</u>
1. <u>Design #1</u>	
2. <u>Design #2</u>	
3. <u>Design #3</u>	
4. <u>Similarities and Differences</u>	
5. <u>Summary of Design Alternatives</u>	
<u>Page # 20 - 21</u>	<u>The Best Design Alternatives</u>
<u>Page # 22 - 27</u>	<u>Preliminary Design & Final Report Draft</u>
1. <u>Level 1 Schematic</u>	
2. <u>Circuit Design</u>	
3. <u>Software Design</u>	
<u>Page # 28 - 49</u>	<u>Testing</u>
<u>Page # 50 - 53</u>	<u>Final Design</u>
<u>Page # 54</u>	<u>Budget</u>
<u>Page # 55 - 61</u>	<u>Timeline</u>
<u>Page # 62 - 64</u>	<u>Contributions</u>
<u>Page # 65-66</u>	<u>Lessons Learned</u>
<u>Page # 67</u>	<u>Conclusion</u>

<u>Page # 68</u>	<u>References</u>
<u>Page # 69 - 75</u>	<u>Appendix A</u>
<u>Page # 76 - 77</u>	<u>Appendix B</u>
<u>Page # 78 - 84</u>	<u>Appendix C</u>

Abstract

According to www.fbi.gov in 2016, more than 460,000 children are reported missing every year. In order to combat this more effectively, we decided to build an AI dashcam using transfer learning to help law enforcement identify cars described in an amber alert. This device is not only useful for this purpose, but also for quickly detaining suspects of human trafficking, grand theft auto, and other crimes involving the use of automobiles. Using the descriptions provided for the amber alert car, the device processes its field of vision and searches for that specific car. Our device uses a camera to take pictures of vehicles in front of it, and processes these images through a neural network to see if a car's features match the suspected traits detailed by the amber alert. At the end of our development cycle, we were able to successfully train our neural network to identify different cars based on their model.

Project Selection and Problem Scoping

1. Projects Goals:

This project is a DASHCAM with AI functionality geared towards identifying specified AMBER/SILVER alert car suspects. A majority of the project involves training machine learning models via transfer learning, such that the dashcam actively scans the field of view using a camera module and searches for potential matches of a specific car. For instance, the officer is notified of the following amber alert on his/her squad car computer: *1997 Honda Del Sol, White*. The officer may load a previously trained model on the dashcam to actively seek that specific car on the road. If a potential match is found, a GPS module in the dashcam will report the coordinates of the potential match, as well as a picture of the potential match. The purpose of this technology is to aid law enforcement in hopes that the victim(s) will be saved, though human trafficking advocacy groups may be interested in using the device as well.

2. Needs Statement:

Driving is a daily part of everyone's life, but it is also very dangerous. According to the National Center for Missing and Exploited Children, "203,000 children are kidnapped each year by family members. Another 58,200 are abducted by non-family members[1]." Besides, by a case study in 2001, "840,279 people (adults and children) were reported missing to the FBI's National Crime Information Center (NCIC).[2]" The number has grown since then, and many people fall victim to human trafficking. With the right information and technology, these tragedies can be prevented.

3. Objective Statement:

The Dashcam should be easy to use and relatively quick to run as amber alerts are emergency situations. It needs to be relatively secure from hackers or unwanted access; it can't be intrusive or block the driver's view and it should be minimal, needs access to a directory to store potential evidence and footage, should not store or save pictures that don't assist in the search and needs to be able to analyze the vehicles to differentiate models; also needs to be able to compare cars with the specific model so that all criteria are met for matching the amber alert.

4. Background:

We will discuss a recent neural network learning method and device patent. This is a patent regarding the implementation of deep learning for the use of building a method meant to replicate human brain neural networks. To explain briefly, the method acquires data, analyzing it in subprocesses called “Masks” in which specific attributes of said data is tested, which results in that particular subset of data attaining a “weight” value which is used as a factor in the overall unit “predicting and classifying” the data of the next input [3].

Convolutional Neural Network is a type of neural network used for many forms of recognition by focusing on specific features of an image at one level, obtaining data, and then advancing to another level, effectively handling the image in layers. Each layer is analyzed as a matrix with each index representing a pixel value. The array is multiplied by a set of weights, all of which are summed, and an activation function is applied to it. (1) “Neural network(s) are composed of many neurons; the output of the previous neuron can be used as the input of the latter neuron. The corresponding formula is as follows:” [4].

$$h_{w,b}(x) = f(W^T x) = f\left(\sum_{i=1}^3 W_i x_i + b\right) \quad (1)$$

This sort of Neural Network is a candidate for what we should use for our car recognition AI model. The reason why convolutional neural networks are the best method to adopt for this project is because CNNs effectively work in specific parts that can be conjoined into a single value. These parts synchronise with specific aspects of our project, for example: a part of the CNN dedicated specifically to discerning cars from other objects, a part of the CNN dedicated to specific classification of a headlight model, and the same idea for front bumpers and so on. Since there are multiple aspects to test for in order for a successful result, convolutional neural networks are a perfect choice for accomplishing this with minimal parameters and competitive efficiency.

Deepface is the software that Facebook uses to recognize and classify faces. They use a common AI pipeline that is as follows: detect -> align -> represent -> classify. In this model,

they focus deeply on the aligning and representing stages “by employing explicit 3D face modeling in order to apply a piecewise affine transformation, and derive a face representation from a nine-layer deep neural network” [5].

Over many news sites like Consumer Report, dashcam are proving to be good personal security like to obtain evidence for insurance claims. These dashcam come equipped in many different ways to connect like wifi, 4g and bluetooth. These dashcam in addition come equipped with storage devices and a camera that record videos in 1080 pixel. Moreover, these dash cam also have motion to detect motion when someone or somebody comes in contact with the cars.

(b) Survey Link:

<https://docs.google.com/forms/d/1fPSXyHimNFRIXFCmXYgtTUutxF4lDo4N-cGKnYykU-k/edit#responses>

Survey Results:

- One of the first questions we asked was what type of information the dashcam should send law enforcement. We gave people 3 options
 - (a.) Picture of the suspected car from the view of the dashcam
 - (b.) GPS Location of the suspected car
 - (c.) Both

[See appendix for Survey image A](#)

- As you can see from our results above, 93% of surveyed individuals believe that the webcam sending both pieces of information is best
- For our next question, we asked if we should include any additional information to the police when the pictures and GPS are sent from the dashcam. Based on our results below, we can see that there was a large variety of information people felt should also be reported to the police: license plate, date, driver identity, etc. All of this feedback will be tremendously useful in designing our project so we can have an idea of what we need

[See appendix for Survey Image B](#)

- The next question we asked was if we should alert the driver of the dashcam that they are near a suspected vehicle. Based on our results below we can see that over 87% of people believed that people should be alerted

[See Appendix for Survey Image C](#)

- Our following question was if they believed our product violated anyone's privacy. All of the responses agreed that this wasn't a violation of privacy since it is all done for safety but we should make sure we let the driver of the dashcam know what information it can record

[See Appendix for Survey Image D](#)

- We then asked what organizations we should be sending this information to. We were pleasantly surprised that people recommended that any information the dashcam record should be sent to anti-human trafficking organizations as well as car dealerships who may encounter these vehicles. This will help us branch out our database network

[See Appendix for Survey Image E](#)

- We asked if there was any preference for product design, in terms of portability, ease of use, the complexity of features, etc. While a decent majority didn't have a preference for features as long it worked as intended, 40% of people wanted the product to be portable and limited functionality as long as it was easy to use

[See Appendix for Survey Image F](#)

- We asked under what price range people would buy our product under. It looks like 61% of people were comfortable with a pricing range of \$50 - \$150. This will help us determine the quality of components we need to use to meet that price

[See Appendix for Survey Image G](#)

- We asked what type of interface people wanted, and it seems that the majority chose a phone app as a favorable interface with 43% of the vote. Although this was the most favorable choice, we ended up going with a touch screen device as it is more safe to use when driving, than to use your phone

[See Appendix for Survey Image H](#)

- We asked what other features our product should do and a majority of people suggested that our product should include the ability to record video and audio, this will prove useful in helping the police gather more evidence to place charges on the suspected vehicle

[See Appendix for Survey Image I](#)

- The last question we asked was if there were any concerns with this product and it seems that the main concerns a majority of people had been in regards to privacy where this

information could be sold to 3rd party companies if pictures are taken then any nearby driver's license plates are also captured in the picture. People also don't want the dashcam's main features to be accessible from the phone since this is a driving distraction.

[See Appendix for Survey Image J](#)

i. Functionality:

The dashcam will send a picture of the suspected car from the view of the dashcam with the location of where the suspected car is found to the police so that the police or any officer can catch up to the suspect car quickly.

ii. Appearance:

The dash cam has to be portable with limited functionality in order to reduce the cost of manufacturing and also have enough space to be near the car front dashboard.

iii. Cost:

The price of the dashcam will be between or around \$50 to \$150 because currently according to the average price of a normal dashcam, it is around \$100 at most.

iv. User Interface:

The dash cam will be intractable with a touch screen so that the user will be able to easily use the camera application and search for a car

v. Reliability:

The dashcam will alert the user when a suspected car is nearby to make sure it is already taking photos, locations and video evidence of the suspected car while also letting the users decide what to do.

vi. Power Requirements:

The dashcam will have a lithium powered battery or power supply directly from an independent power source so that whenever there is no power supply, it will still function and check for amber alert suspected cars if alerted.

vii. Expected Product Lifetime:

The dashcam expected life will be 5 years because that is the expected life expectancy of most dashcam.

viii. Interfacing requirements with other systems:

The dashcam will need an internet connection to send pictures and GPS coordinates to the police station and should be able to stay behind the driver's windshield and remain stationary in order to get clear photo and video evidence and send and notify the police.

ix. User training needs:

There will be instructions manual and there will be labels and warnings on what the device will do so that the user can immediately get used to the dashcam and understand what it does.

x. Government regulations and licensing:

The government has been allowing dashcam in order to help take evidence when accidents happen in order to take a long procedure to solve who is right or wrong in the accident.

xi. Industry standards:

The dashcam will be made to be portable and does not use a huge amount of data memory, energy, and space in order to be energy efficient and transfer from one car to another.

xii. Safety issues:

The dashcam will be made sure that it is not taking pictures of the wrong cars and will not be shaped in a sharp shape to prevent any injuries when the device dropped or broke as no innocent civilians will get into a situation when they are not.

xiii. Environmental Concerns:

The device will be made to be recyclable so that it will not be also a part of the pollution.

5. Marketing Requirements:

The dashcam has a couple marketing requirements to fulfill in order to meet the standard quality we are looking for. Here we introduce those said requirements. In order for the dashcam to be successful and meet our standards, it must fulfill these requirements:

Needs to be cost effective.

Needs to be secure. (Information and data will not be leaked out)

Needs to be profitable.

Needs to be novel, something new enough to be a force in the market.

Needs to be attractive.

Should appeal to multiple groups in order to be successful. (The device should work that suits any type of vehicle.)

Should have a good deal of support. (New application developers, debug team, maintenance etc.)

6. Objective Tree:

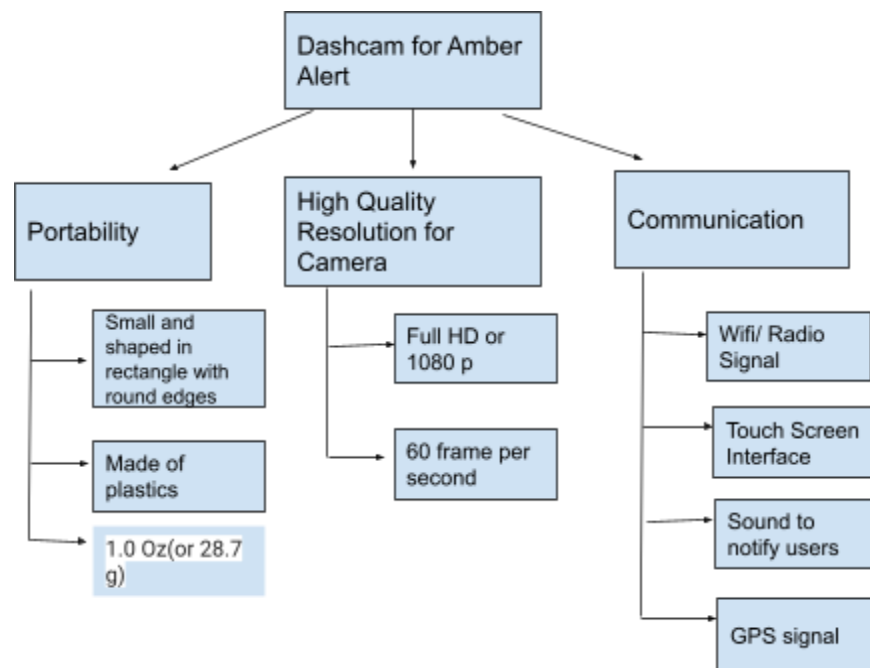


Fig. 1. This figure describes the general milestones to achieve before the dashcam is completed.

Engineering Requirements

1. Functionality

- a. The dashcam will be searching for the identified car and taking picture and video evidence when the amber alert is being notified in the area of the user.

2. Performance

- a. The front camera will have a 1080 p resolution to take clear and smooth picture and video evidence of the suspected car. The AI will be using Convolutional Neural Networks to quickly identify a specific car. Feeding many pictures of the same car to the AI should increase the accuracy.

3. Economic

- a. The device needs to be low cost in order for law enforcement to consider utilizing it in each squad car.

4. Energy/Power

- a. This Dashcam should be powered through a 24W 4.8A car charger (5V)

5. Environmental

- a. N/A. Justification: This device has no environmental requirements because the device by nature does not harm the environment, as it is electrically powered.

6. Health & Safety

- a. Dashcam will be 3 inches wide and 2 inches tall so that it takes up minimal space
- b. Should be able to be attached to either the windshield or dashboard as to avoid blocking the driver's view

7. Legal

- a. Should not breach the privacy of the user

8. Maintainability

- a. Should be chargeable.

9. Manufacturability

- a. The Device will be made out of plastic, metal, and glass. Plastic for the exterior of the product. Metal for the interior hardware components, and glass for the camera

and touch screen. All of this will be able to be manufactured at factories since we're not using special material

10. Operational

- a. Should be operated by the user via touchscreen, and should have common hardware functions (Power, Brightness, Send Data, etc.)

11. Political

- a. The dashcam is specially designed for the law enforcement to use and is not concerned with the politic as it only serves as evidence searching and taking device so that it takes less time and Human Resources to search for the amber alert car.

12. Reliability and Availability

- a. Devices will be secured to ensure no data is leaked to 3rd parties. Products will be available at all physical and digital retailers that sell electronics. All images that do not match the vehicle description will be deleted from the Jetson's Nano storage

13. Social and Cultural

- a. N/A. There are no Social and Cultural requirements since this device is meant to aid law enforcement and is not within the Social and Cultural realm.

14. Usability (including user-interface)

- a. The dashcam will be using the touchscreen in order to allow the user to directly interact and use the functions of the device without having to cause disturbance when driving like taking out phones just to use the dashcam and it will also be able to attach to the car dashboard or the front windshield.

Table I

Marketing Requirements	Engineering Requirements	Justification
4, 6	1.) The dashcam should scan its field of vision to differentiate cars from other objects.	This is novel and new to the market because no other dashcam has been integrated to work with amber alerts and gather evidence for the police.
3, 4, 6, 7	2.) The front camera will have a 1080 p resolution to take clear, and smooth pictures and video evidence of the suspected car.	Typically dashcams don't take pictures of specific cars if they aren't involved in an accident so this is a new idea.
4	3.) The dashcam should pass the second level of the AI implementation, identifying specific cars with at least 90% accuracy.	The AI will be used for the new functionality of recognizing cars accurately to make sure only the suspected car is being recorded for evidence.
1, 3, 5, 6	4.) Dashcam will be 3 inches wide and 2 inches tall so that it takes up minimal space.	Making sure it is compact ensures that it will only have components essential to its main functions. Making it this compact will ensure that it doesn't impede the driver's view
5	5.) Should be able to be attached to either the windshield or dashboard as to avoid blocking the driver's view	To avoid disturbance of the driver, the device will be able to attach to places on the car so that it can still take evidence and at the same time prevent obstruction of the driver's view.
5, 6	6.) The user interface should be responsive and easy to use.	To allow the user to immediately and directly use the device functions, unlike indirect methods.
2, 7	7.) The device and the evidence stored will be secured and if the images of any vehicles to not match the suspected	Ensuring that all evidence gathered is secure will give both the customer and law enforcement confidence in our product and put our customers at

	description, then they will be deleted	ease knowing that their information isn't at risk.
4, 7	8.) Evidence such as pictures, GPS location, and video should be clear to understand.	To directly and immediately send evidence so that law enforcement can take action immediately and quickly.

Table I correlates marketing requirements from Table II to different engineering requirements of our project and how our project justifies both requirements

Table II

<p>Marketing Requirements List:</p> <ol style="list-style-type: none"> 1. Needs to be cost-effective. 2. Needs to be secure. (Information and data will not be leaked out) 3. Needs to be profitable. 4. Needs to be novel, something new enough to be a force in the market. 5. Needs to be attractive/ convenient. 6. Should appeal to multiple groups in order to be successful. (The device should work that suits any type of vehicle.) 7. Should have a good deal of support. (New application developers, debug team, maintenance etc.)
--

Table II provides the list of features we believe customers will look for when looking at our product

Design Alternatives

Three Design Alternatives:

Table III

	Location	Communication	AI Implementation	Embedded System
Alternative 1	GPS module	Internet Hotspot	Processed in Microcontroller	NVidia Jetson
Alternative 2	Internet Controlled	Radio Waves	Processed in server	Tiva C
Alternative 3	Cellular signal	Phone Carrier	Image/GPS -> M.C Matching -> server	Raspberry Pi

Table III lists the different product designs that were considered

Out of all the alternatives, we decided to go on with alternative 1.

1. Design #1

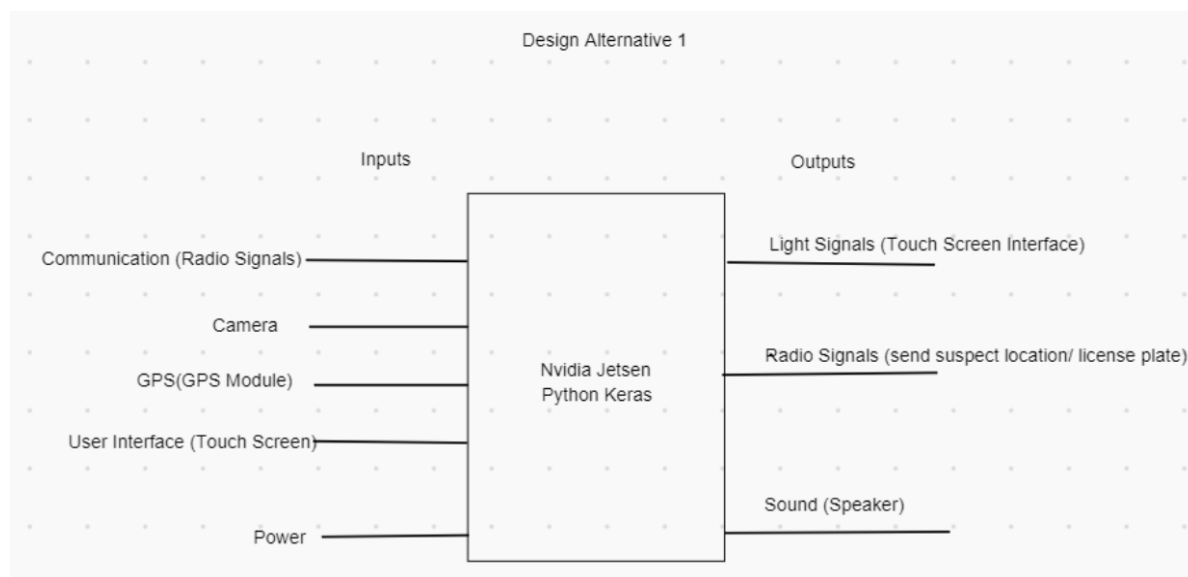


Fig. 1. Depicts design alternative 1.

2. Design #2

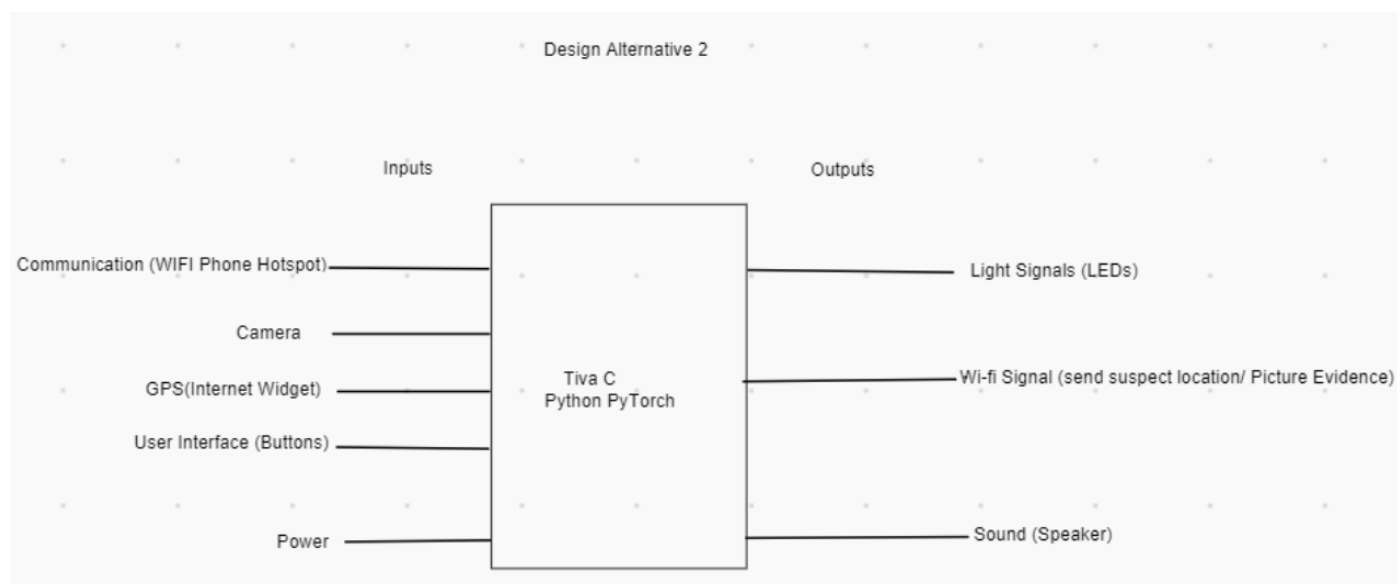


Fig. 2. Depicts design alternative 2.

3. Design #3

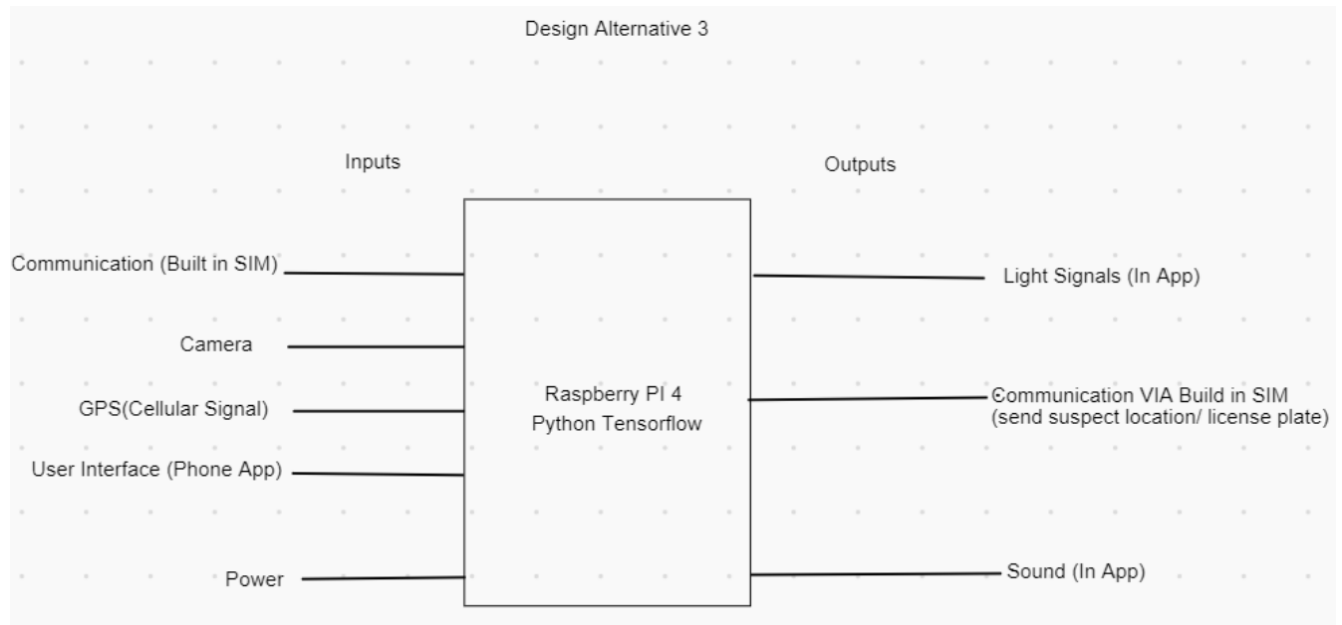


Fig. 3. Depicts design alternative 3.

4. Similarities and Differences

Table IV

Similarities	Differences
Output Data Type (GPS coordinates, image)	Sequence of operations:
Camera is attached to the dashcam	Communication method: Radio, Internet Hotspot, or Cellular
Usage of microcontroller for image processing	Usage of different boards: Tiva C, NVIDIA Jetson, or RaspberryPi 4
Amber alert signals are received by the dashcam	Method of user interaction: Touch screen on the device

Utilizing Python for AI	Which Python Libraries: Keras, Tensorflow, Pytorch
GPS is used to validate the Amber Alert range	Location method: GPS, WIFI Hotspot, or Cellular
Images are processed and analyzed for the model of cars	
Data is sent to the police	

Table IV lists the similarities all of our designs have in common and on the right side, all of their differences are listed

5. Summary of Design Alternatives

We conclude that each design alternative maintains certain key components and features. Each design alternative proposes the following: Camera usage, Embedded system for AI processing, the sending and receiving of data to and from a server for police functions, and Power supplied from car batteries. The following present the key differences among these design alternatives:

Regarding Location Functionality:

- GPS (Alternative 1)
- Internet Widget (Alternative 2)
- Cellular Signal (Alternative 3)

Embedded System:

- NVidia Jetson (Alternative 1)
- Tiva C (Alternative 2)
- Raspberry PI 4 (Alternative 3)

Python Library:

- Tensorflow (Alternative 1)
- Google Keras (Alternative 2)
- PyTorch (Alternative 3)

User Interface:

- Touch Screen (Alternative 1)
- Buttons (Alternative 2)
- Phone App (Alternative 3)

Table V

Level 0			
Module	Design #1	Design #2	Design #3
Inputs	Radio Waves, GPS Signal	Wifi Signal, Gps Signal	Cellular Signal, GPS signal
Outputs	Radio Waves	Wifi Signal	Cellular Signal
Functionality	Send and receive notifications to and from the police about the amber alert pictures evidence and also the location.		

Table V displays the different input and output methods of our 3 designs

The Best Design Alternative

Criteria:

- 1. Your group's technical knowledge about the design alternative**
 - We need to be able to program and code the AI so we must ensure that the device we choose uses a language we're familiar with and allows us to easily implement external devices such as a camera or Wi-Fi adapter
- 2. Availability of any required datasheets, application, notes, or development kits**
 - We have to make sure that our project can be built and that all the parts that we needed are available. And in conclusion, we have found all the parts we need in order to complete our project.
- 3. Suitability for demo for 397**
 - To demonstrate for 397 we could give our device an image of a car to process and see if it matches with a mock amber alert description
- 4. Ability to manufacture**

- Our device uses an Nvidia Jetson, a camera, and dashcam shell which are all already available to purchase so we can manufacture them easily

5. Ease of product use

- The device will be easy to use with functions built into it using the touch screen on the LCD screen

Pairwise Comparison of the selection criteria

- 1) [Eric](#)
- 2) [Sai](#)
- 3) [Minh](#)
- 4) [Sean](#)

Table of Identification and rate of alternatives relative to the criteria

Compute scores for alternatives

Table VI

		Alternative 1	Alternative 2	Alternative 3
Criteria 1	0.511	5	3	3
Criteria 2	0.26	5	5	5
Criteria 3	0.128	3	5	5
Criteria 4	0.0625	3	5	1
Criteria 5	0.032	5	3	3
Score		4.587	3.882	3.632

Table VI displays the rankings Eric gave to all of the project designs

Justification

We believe that alternative 1 is the best choice because since it uses the Nvidia Jetson it is more feasible for us to accomplish this project using our technical knowledge. In terms of development kits, the Nvidia Jetson and microcontroller are relatively cheap and easy to purchase so availability will not be an issue. These components are physically small enough for us to perform a demonstration for ECE 397 without any issues regarding portability. Nvidia Jetson, micro controller, and its camera are all easily purchasable from their respective websites so there won't be any issues regarding manufacturing. In terms of the ease of product use, we believe that programming with the Nvidia Jetson will allow us to create a clean user interface to ensure the product is easy to use for potential users.

Preliminary Design & Final Report Draft

- **Level 1 Schematic:**

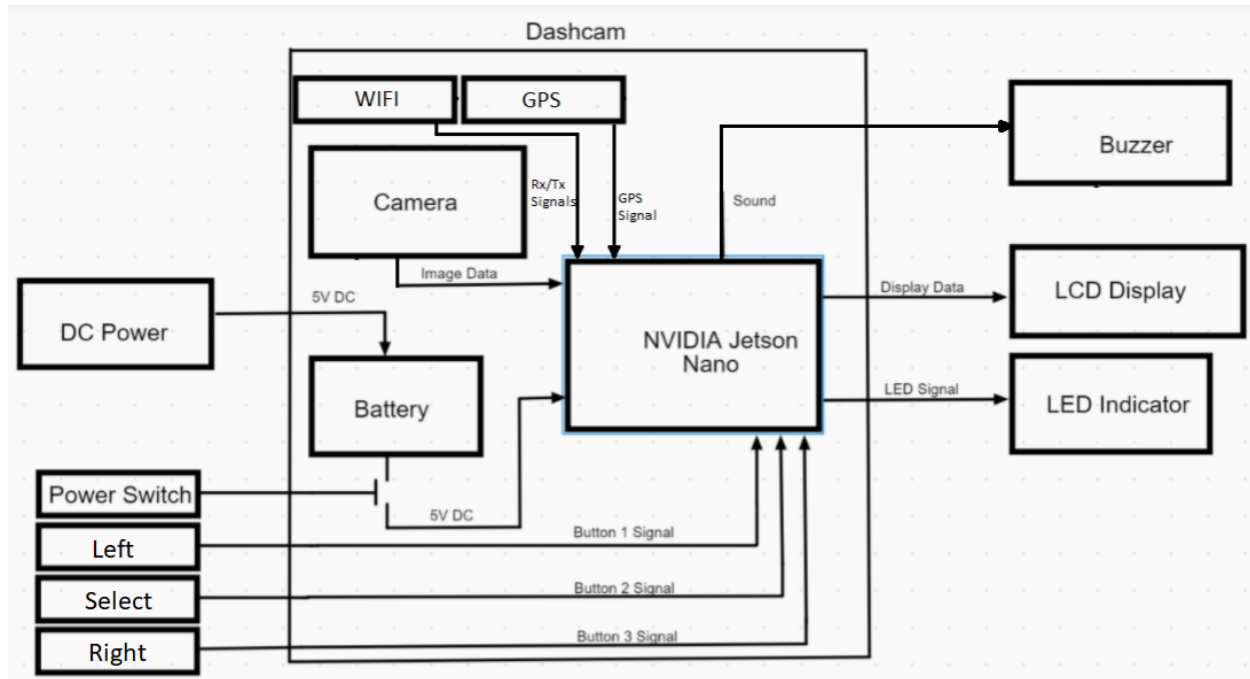


Fig. 1. Depicts the Level 1 schematic of the device.

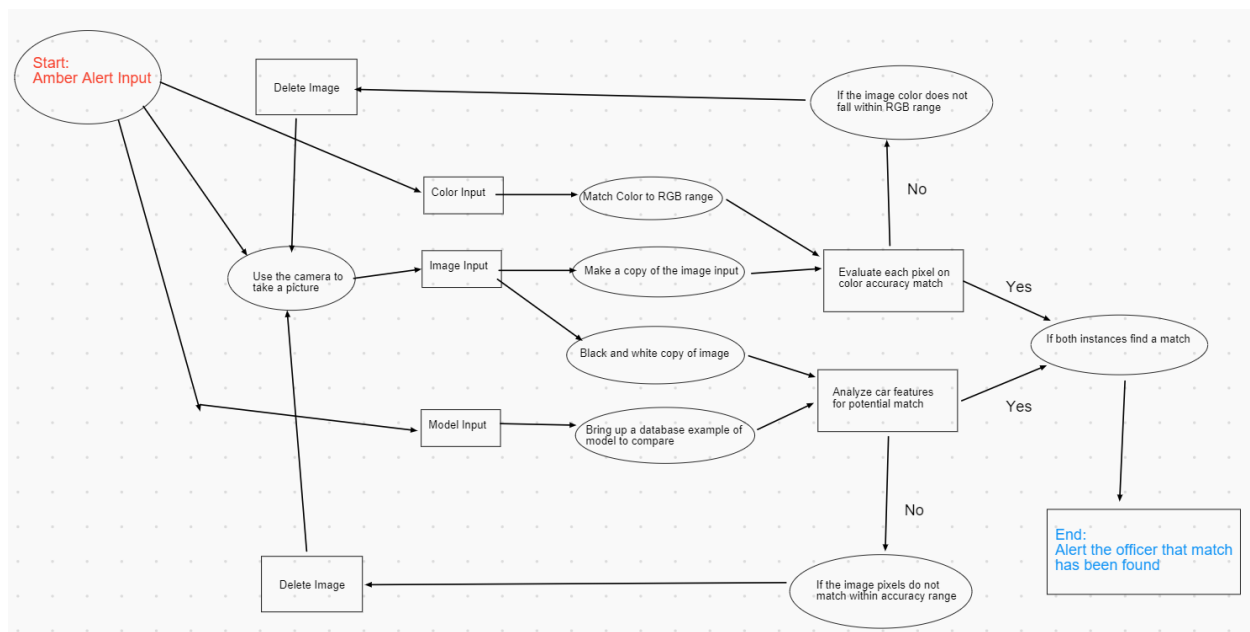


Fig. 2. Depicts a flow chart of the general software operations of the device.

- **Circuit Design:**

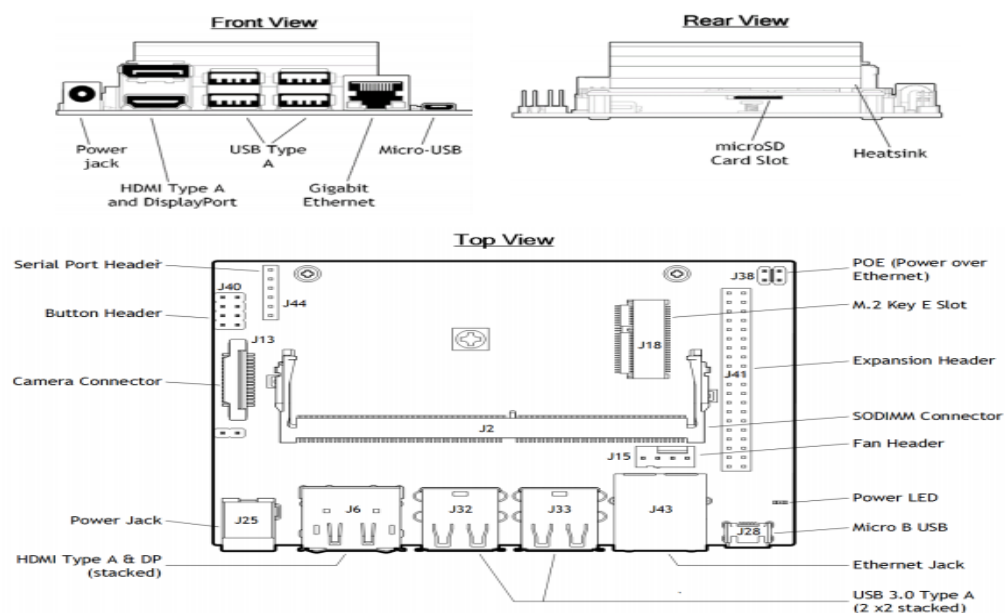


Fig. 3. Depicts NVIDIA's Jetson Nano embedded structure diagram. [6]

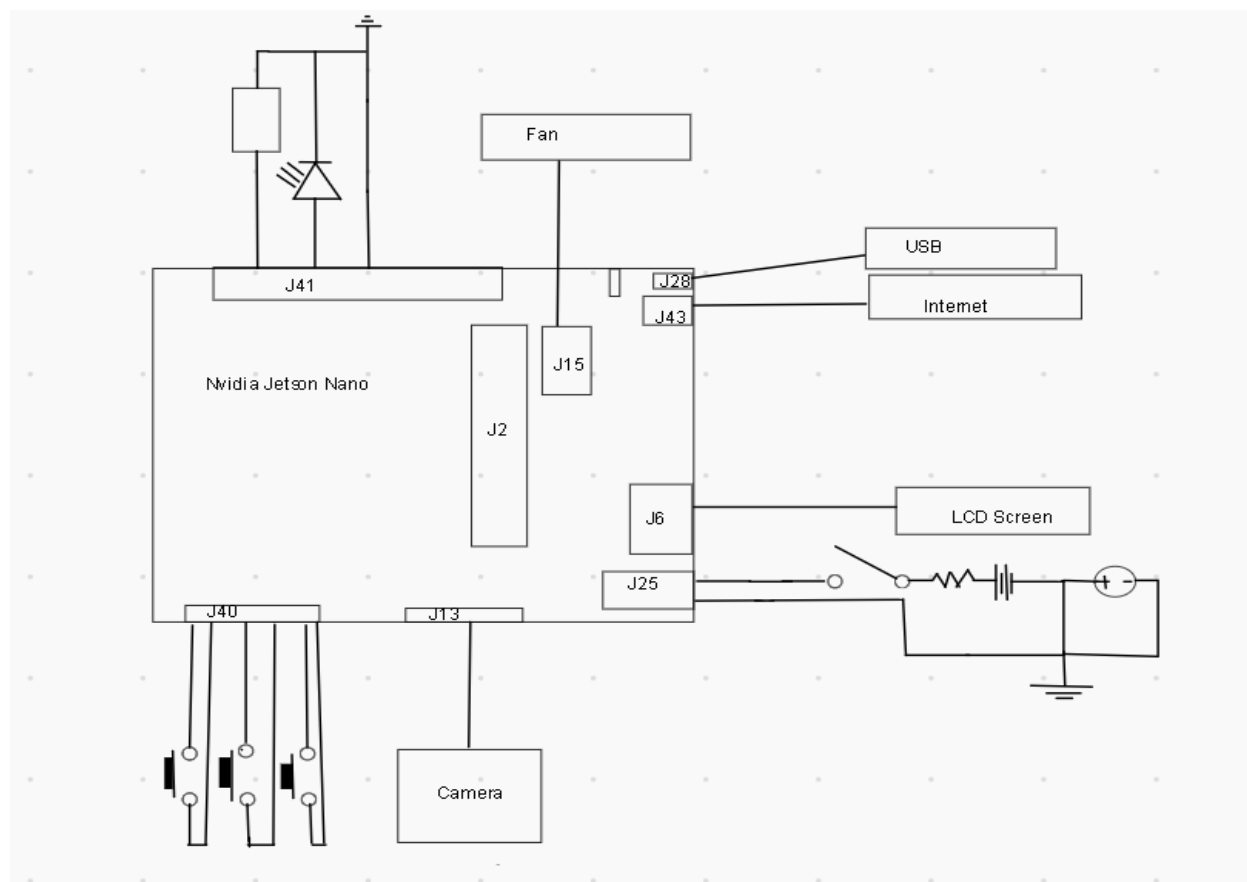


Fig. 4. Depicts module integration for each component.

- **Software Design:**

Transfer Learning

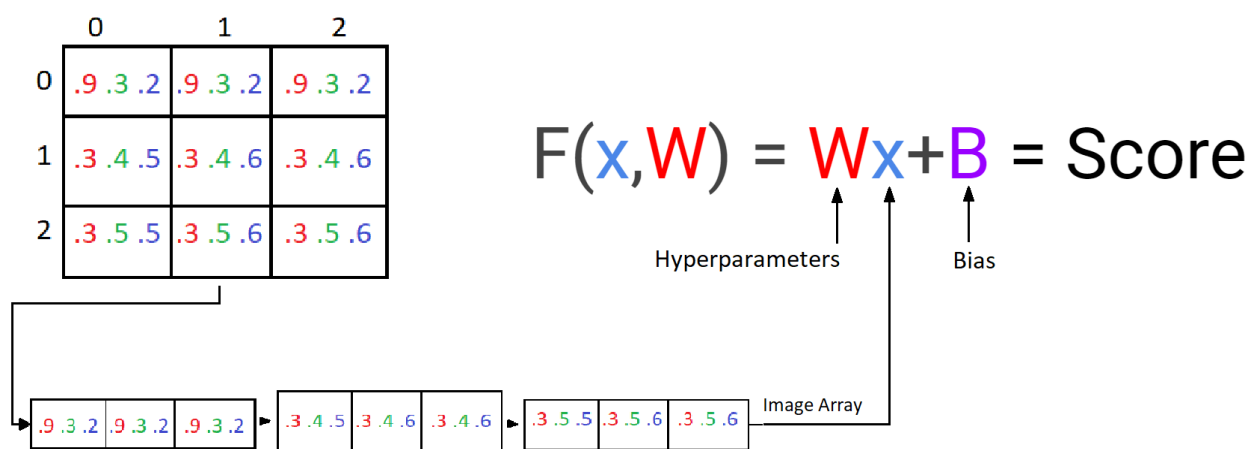


Fig. 5. Depicts linear classifier operations- used in convolutional neural networks.

We will be using Transfer Learning to train the device. Transfer learning is a method for training models when limited data is available. The last few layers of Convolutional Neural Networks are retrained for the purpose of classifying data. These networks consist of linear classifiers, which are regularly updated as the model is trained with new data. In the figure above, x represents the image array, which is an n by m by 3 two dimensional array with n being the width, m being the height, and 3 being a constant representing RGB values. B is the bias value that is adjusted depending on the data set used to train the algorithm. This value may be higher if the algorithm is trained with a data set that contains more of an object x as opposed to an object y . W represents the hyperparameters, which comprise of the hidden layers within the algorithm. These values are untouched by the programmer, as it is the duty of the program to select hyperparameters that will ultimately result in more accurate classification abilities. A score is assigned by the linear classifier, and if the classifier guesses correctly, the hyperparameters are adjusted accordingly and vice versa. [7]

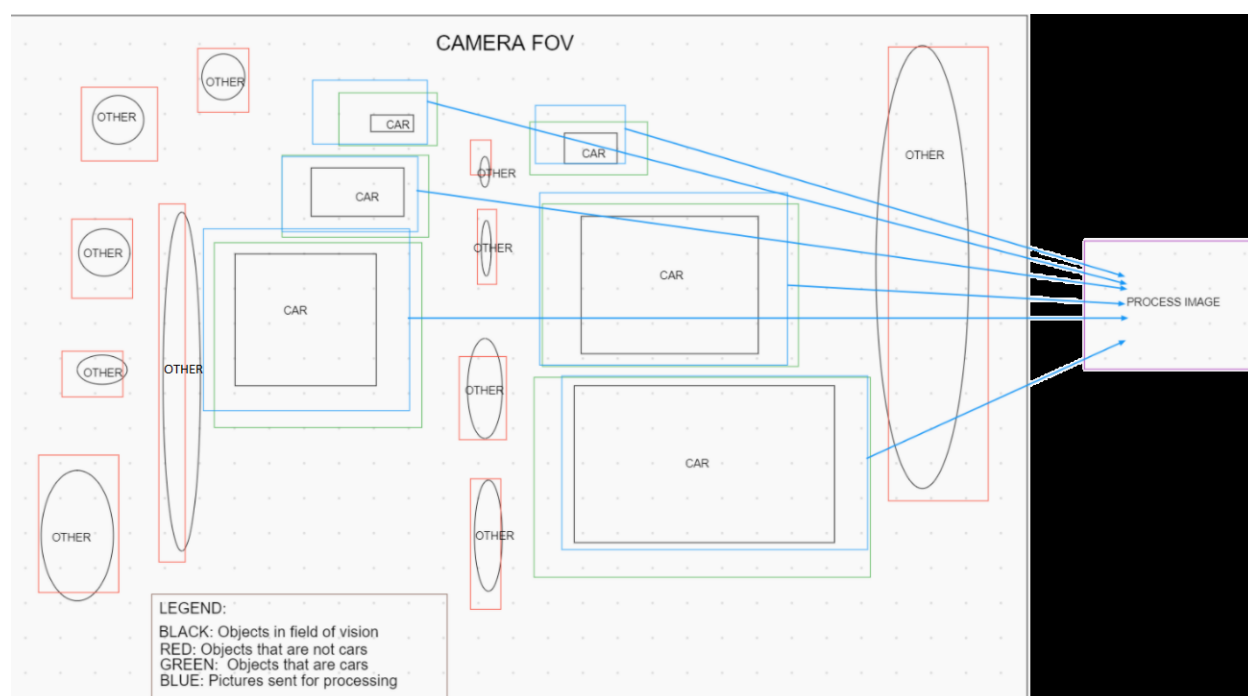


Fig. 6. Depicts how the device should only handle objects in the FOV that are cars.

The above figure depicts the first test for the neural network, which is to distinguish cars from different objects in the field of view. Every car detected in the image will be cropped and sent for image processing and tested by other classifiers for feature detection. This is accomplished by the first linear classifier, which is trained using a data set of images of random cars, such that it will be able to recognize if there is a car in any image. This cycle of detecting cars and cropping sub images to be analyzed is a process that will repeat every 10 seconds elapsed since the previous set.

Pseudo Code: (Project will be done in Python)**(Primary)**

Check if object is a car

bool Fcar(x,W);

While(car)

- Check if front or back of car
 - Forien(x,W);
- Check each feature (if front)
 - FheadLight(x,W);
 - Fbumper1(x,W);
 - FwindShield(x,W);
 - Fgrill(x,W);
- Check each feature (if back)
 - FtailLight(x,W);
 - Fbumper2(x,W);
 - Ftrunk(x,W);
- Convolve results into confidence value
 - if(back)
 - confidence(FtailLight(x,W), Fbumper2(x,W),Ftrunk(x,W));
 - if(front)
 - confidence(FheadLight(x,W), Fbumper1(x,W),FwindShield(x,W), Fgrill(x,W));
- Check confidence value $\geq 94\%$
 - if(match)
 - send(GPS());
 - if(!match)
 - deleteAll();

(Secondary)

boolean amberAlert;

color; // string from amber alert

feature; // car model string from amber alert

int fullMatch = 0;

def takePicture(){

return nvidiaJetsonCamera;

// use the camera module to take a picture

}

```

def featureMatch(feature, compare) {
    Test specific feature for match          // function is called for each feature
    If match return true, false otherwise    // (windshield, hood to fender ratio etc)
}
def colorCode(color) {
    if(color == "red"){                      // takes in a color and returns that color's RGB values
        return [255, 0, 0];
    }
    ...
}
def colorMatch(img, color){
    colorRGB = colorCode(color)
    width, height = img.size;
    while (i = 0; i < length; ++i){
        if (j = 0; j < width; ++j){
            if ((img[i,j] > (colorRGB - 30))    // check if pixel color of the image
                && (img[i,j] > (colorRGB + 30))) // matches the color of amber alert
            }
            else {
                return FALSE;
            }
        }
    }
    return TRUE;
}
def match(amberAlert, color, feature, compare){
    while (amberAlert == TRUE){              // As long as the amber alert is active, keep running
        image = Image.open(takePicture());   // Store the image taken from Jetson
        imageBW = image;                     // Make a copy of said image
        imageBW = imageBW.convert('1');      // Make the copy black and white
        if(colorMatch(image) &&
            bwMatch(imageBW) &&
            featureMatch(feature, compare)){ // Do and return if all match
            return;
        }
    }
}
def feedForward(dataset){
    Train the network using linear classifiers,
    Update compare variables
}

```

Testing:**Confusion Matrix:*****Objective:***

In this test, we monitor the accuracy of the model with four different cars, each tested against five different photos per car, trained at various epochs. We use the following confusion matrix to determine which cars the model predicts correctly. The values in the figure below indicate a perfect score, representing an effectively trained model. We tested to see how many epochs was sufficient to achieve a perfect score.

Confusion Matrix				
True Label	Acura	Aston	BMW	Dodge
	5	0	0	0
	0	5	0	0
	0	0	5	0
	0	0	0	5
Predicted Label				
	Acura	Aston	BMW	Dodge

Fig. 1. Shows the confusion matrix where the neural network displays the accuracy of its classification of 4 different car images in 4 different classes

Method:

We increase the amount of epochs as well as note the model's accuracy per trial until a perfect score is achieved.

Observations:***1 Epoch:***

Epoch 1/1

24/24 - 5s - loss: 1.3393 - accuracy: 0.4917 - val_loss: 0.9629 - val_accuracy: 0.6667

[[1 3 0 1]

[1 4 0 0]

[2 1 2 0]

[0 0 0 5]]

2 Epochs:

Epoch 1/2

24/24 - 5s - loss: 1.2364 - accuracy: 0.5292 - val_loss: 2.7937 - val_accuracy: 0.2000

Epoch 2/2

24/24 - 5s - loss: 0.2564 - accuracy: 0.9375 - val_loss: 0.8810 - val_accuracy: 0.6333

[[2 0 0 3]

[0 2 0 3]

[1 0 1 3]

[0 0 0 5]]

3 Epochs:

Epoch 1/3

24/24 - 5s - loss: 1.2792 - accuracy: 0.5375 - val_loss: 0.9162 - val_accuracy: 0.6667

Epoch 2/3

24/24 - 5s - loss: 0.2438 - accuracy: 0.9417 - val_loss: 0.5499 - val_accuracy: 0.7667

Epoch 3/3

24/24 - 5s - loss: 0.0943 - accuracy: 0.9958 - val_loss: 0.5727 - val_accuracy: 0.7000

[[5 0 0 0]

[0 4 1 0]

[2 0 3 0]

[0 0 0 5]]

4 Epochs:

Epoch 1/4

24/24 - 5s - loss: 1.0210 - accuracy: 0.6000 - val_loss: 0.6034 - val_accuracy: 0.7667

Epoch 2/4

24/24 - 5s - loss: 0.2486 - accuracy: 0.9458 - val_loss: 0.4568 - val_accuracy: 0.8667

Epoch 3/4

24/24 - 5s - loss: 0.1010 - accuracy: 1.0000 - val_loss: 0.4537 - val_accuracy: 0.8333

Epoch 4/4

24/24 - 5s - loss: 0.0637 - accuracy: 1.0000 - val_loss: 0.2470 - val_accuracy: 0.9333

[[5 0 0 0]

[0 5 0 0]

[2 0 3 0]

[0 0 0 5]]

5 Epochs:

Epoch 1/5

24/24 - 5s - loss: 1.3094 - accuracy: 0.4917 - val_loss: 1.6369 - val_accuracy: 0.5000

Epoch 2/5

24/24 - 5s - loss: 0.2649 - accuracy: 0.9458 - val_loss: 0.8328 - val_accuracy: 0.6333

Epoch 3/5

24/24 - 5s - loss: 0.1437 - accuracy: 0.9792 - val_loss: 0.8560 - val_accuracy: 0.7000

Epoch 4/5

24/24 - 5s - loss: 0.0909 - accuracy: 0.9958 - val_loss: 0.6679 - val_accuracy: 0.8000

Epoch 5/5

24/24 - 5s - loss: 0.0420 - accuracy: 1.0000 - val_loss: 0.6077 - val_accuracy: 0.7000

[[5 0 0 0]

[1 4 0 0]

[3 0 2 0]

[0 0 0 5]]

6 Epochs:

Epoch 1/6

24/24 - 6s - loss: 1.0403 - accuracy: 0.5625 - val_loss: 1.4252 - val_accuracy: 0.5000

Epoch 2/6

24/24 - 5s - loss: 0.2207 - accuracy: 0.9625 - val_loss: 1.4606 - val_accuracy: 0.4333

Epoch 3/6

24/24 - 5s - loss: 0.0767 - accuracy: 1.0000 - val_loss: 0.6521 - val_accuracy: 0.6667

Epoch 4/6

24/24 - 5s - loss: 0.0631 - accuracy: 1.0000 - val_loss: 0.3984 - val_accuracy: 0.8333

Epoch 5/6

24/24 - 5s - loss: 0.0485 - accuracy: 1.0000 - val_loss: 0.4937 - val_accuracy: 0.8667

Epoch 6/6

24/24 - 5s - loss: 0.0275 - accuracy: 1.0000 - val_loss: 0.3642 - val_accuracy: 0.8667

[[5 0 0 0]

[0 5 0 0]

[2 0 3 0]

[0 0 0 5]]

7 Epochs:

Epoch 1/7

24/24 - 5s - loss: 0.9975 - accuracy: 0.5667 - val_loss: 0.7787 - val_accuracy: 0.6667

Epoch 2/7

24/24 - 5s - loss: 0.2680 - accuracy: 0.9375 - val_loss: 0.6611 - val_accuracy: 0.8000

Epoch 3/7

24/24 - 5s - loss: 0.1163 - accuracy: 0.9958 - val_loss: 0.5380 - val_accuracy: 0.7667

Epoch 4/7

24/24 - 5s - loss: 0.0586 - accuracy: 1.0000 - val_loss: 0.3280 - val_accuracy: 0.9000

Epoch 5/7

24/24 - 5s - loss: 0.0376 - accuracy: 1.0000 - val_loss: 0.2676 - val_accuracy: 0.9333

Epoch 6/7

24/24 - 5s - loss: 0.0346 - accuracy: 1.0000 - val_loss: 0.2264 - val_accuracy: 0.9000

Epoch 7/7

24/24 - 5s - loss: 0.0341 - accuracy: 1.0000 - val_loss: 0.2897 - val_accuracy: 0.8667

[[4 1 0 0]

[0 5 0 0]

[1 0 4 0]

[0 0 0 5]]

8 Epochs:

Epoch 1/8

24/24 - 5s - loss: 1.1940 - accuracy: 0.5250 - val_loss: 0.7142 - val_accuracy: 0.6667

Epoch 2/8

24/24 - 5s - loss: 0.2885 - accuracy: 0.9250 - val_loss: 0.5230 - val_accuracy: 0.8000

Epoch 3/8

24/24 - 5s - loss: 0.1216 - accuracy: 0.9833 - val_loss: 0.2954 - val_accuracy: 0.8667

Epoch 4/8

24/24 - 5s - loss: 0.0792 - accuracy: 1.0000 - val_loss: 0.3391 - val_accuracy: 0.9000

Epoch 5/8

24/24 - 5s - loss: 0.0605 - accuracy: 0.9958 - val_loss: 0.1276 - val_accuracy: 1.0000

Epoch 6/8

24/24 - 5s - loss: 0.0304 - accuracy: 1.0000 - val_loss: 0.1739 - val_accuracy: 0.9667

Epoch 7/8

24/24 - 5s - loss: 0.0249 - accuracy: 1.0000 - val_loss: 0.1681 - val_accuracy: 0.9667

Epoch 8/8

24/24 - 5s - loss: 0.0331 - accuracy: 1.0000 - val_loss: 0.2054 - val_accuracy: 0.9000

[[3 2 0 0]

[0 5 0 0]

[2 0 3 0]

[0 0 0 5]]

9 Epochs:

Epoch 1/9

24/24 - 5s - loss: 1.1693 - accuracy: 0.5250 - val_loss: 0.7450 - val_accuracy: 0.7667

Epoch 2/9

24/24 - 5s - loss: 0.2370 - accuracy: 0.9417 - val_loss: 0.5487 - val_accuracy: 0.7667

Epoch 3/9

24/24 - 5s - loss: 0.0998 - accuracy: 1.0000 - val_loss: 0.4897 - val_accuracy: 0.7000

Epoch 4/9

24/24 - 5s - loss: 0.0747 - accuracy: 0.9958 - val_loss: 0.4702 - val_accuracy: 0.7333

Epoch 5/9

24/24 - 5s - loss: 0.0499 - accuracy: 1.0000 - val_loss: 0.3398 - val_accuracy: 0.8333

Epoch 6/9

24/24 - 5s - loss: 0.0406 - accuracy: 1.0000 - val_loss: 0.3308 - val_accuracy: 0.8333

Epoch 7/9

24/24 - 5s - loss: 0.0569 - accuracy: 1.0000 - val_loss: 0.4087 - val_accuracy: 0.8333

Epoch 8/9

24/24 - 5s - loss: 0.0273 - accuracy: 1.0000 - val_loss: 0.2970 - val_accuracy: 0.8667

Epoch 9/9

24/24 - 5s - loss: 0.0257 - accuracy: 1.0000 - val_loss: 0.2183 - val_accuracy: 0.9000

```
[[4 1 0 0]
 [0 5 0 0]
 [1 0 4 0]
 [0 0 0 5]]
```

10 Epochs:

Epoch 1/10

24/24 - 5s - loss: 1.0811 - accuracy: 0.5625 - val_loss: 1.2348 - val_accuracy: 0.4333

Epoch 2/10

24/24 - 5s - loss: 0.2368 - accuracy: 0.9500 - val_loss: 0.6103 - val_accuracy: 0.7333

Epoch 3/10

24/24 - 5s - loss: 0.1289 - accuracy: 0.9917 - val_loss: 0.5478 - val_accuracy: 0.8000

Epoch 4/10

24/24 - 5s - loss: 0.0766 - accuracy: 0.9917 - val_loss: 0.3957 - val_accuracy: 0.8000

Epoch 5/10

24/24 - 5s - loss: 0.0510 - accuracy: 1.0000 - val_loss: 0.2856 - val_accuracy: 0.9333

Epoch 6/10

24/24 - 5s - loss: 0.0344 - accuracy: 0.9958 - val_loss: 0.3069 - val_accuracy: 0.8333

Epoch 7/10

24/24 - 5s - loss: 0.0306 - accuracy: 1.0000 - val_loss: 0.3086 - val_accuracy: 0.8667

Epoch 8/10

24/24 - 5s - loss: 0.0247 - accuracy: 1.0000 - val_loss: 0.3049 - val_accuracy: 0.9333

Epoch 9/10

24/24 - 5s - loss: 0.0177 - accuracy: 1.0000 - val_loss: 0.1798 - val_accuracy: 0.9667

Epoch 10/10

24/24 - 5s - loss: 0.0170 - accuracy: 1.0000 - val_loss: 0.2727 - val_accuracy: 0.9333

```
[[5 0 0 0]
 [0 4 1 0]
 [1 0 4 0]
 [0 0 0 5]]
```

11 Epochs:

Epoch 1/11

24/24 - 5s - loss: 1.1404 - accuracy: 0.5417 - val_loss: 0.8437 - val_accuracy: 0.7333

Epoch 2/11

24/24 - 5s - loss: 0.2463 - accuracy: 0.9458 - val_loss: 0.6156 - val_accuracy: 0.7667

Epoch 3/11

24/24 - 5s - loss: 0.1448 - accuracy: 0.9792 - val_loss: 0.3410 - val_accuracy: 0.8667
 Epoch 4/11
 24/24 - 5s - loss: 0.0629 - accuracy: 1.0000 - val_loss: 0.1868 - val_accuracy: 1.0000
 Epoch 5/11
 24/24 - 5s - loss: 0.0428 - accuracy: 1.0000 - val_loss: 0.2863 - val_accuracy: 0.9000
 Epoch 6/11
 24/24 - 5s - loss: 0.0440 - accuracy: 1.0000 - val_loss: 0.3047 - val_accuracy: 0.8667
 Epoch 7/11
 24/24 - 5s - loss: 0.0286 - accuracy: 1.0000 - val_loss: 0.3516 - val_accuracy: 0.8667
 Epoch 8/11
 24/24 - 5s - loss: 0.0224 - accuracy: 1.0000 - val_loss: 0.2125 - val_accuracy: 0.9333
 Epoch 9/11
 24/24 - 5s - loss: 0.0222 - accuracy: 1.0000 - val_loss: 0.1493 - val_accuracy: 0.9667
 Epoch 10/11
 24/24 - 5s - loss: 0.0161 - accuracy: 1.0000 - val_loss: 0.2355 - val_accuracy: 0.9000
 Epoch 11/11
 24/24 - 5s - loss: 0.0179 - accuracy: 1.0000 - val_loss: 0.2604 - val_accuracy: 0.9000

```
[[5 0 0 0]
 [0 5 0 0]
 [1 0 4 0]
 [0 0 0 5]]
```

12 Epochs:

Epoch 1/12
 24/24 - 5s - loss: 1.0043 - accuracy: 0.5667 - val_loss: 0.7664 - val_accuracy: 0.7333
 Epoch 2/12
 24/24 - 5s - loss: 0.2423 - accuracy: 0.9500 - val_loss: 0.6176 - val_accuracy: 0.7667
 Epoch 3/12
 24/24 - 5s - loss: 0.1083 - accuracy: 0.9833 - val_loss: 0.4789 - val_accuracy: 0.8333
 Epoch 4/12
 24/24 - 5s - loss: 0.0756 - accuracy: 0.9958 - val_loss: 0.2600 - val_accuracy: 0.9000
 Epoch 5/12
 24/24 - 5s - loss: 0.0394 - accuracy: 1.0000 - val_loss: 0.3131 - val_accuracy: 0.8667
 Epoch 6/12
 24/24 - 5s - loss: 0.0381 - accuracy: 0.9958 - val_loss: 0.3381 - val_accuracy: 0.8333
 Epoch 7/12

24/24 - 5s - loss: 0.0252 - accuracy: 1.0000 - val_loss: 0.2963 - val_accuracy: 0.8667
 Epoch 8/12
 24/24 - 5s - loss: 0.0389 - accuracy: 1.0000 - val_loss: 0.2463 - val_accuracy: 0.9000
 Epoch 9/12
 24/24 - 5s - loss: 0.0210 - accuracy: 1.0000 - val_loss: 0.2656 - val_accuracy: 0.8667
 Epoch 10/12
 24/24 - 5s - loss: 0.0193 - accuracy: 1.0000 - val_loss: 0.2149 - val_accuracy: 0.8667
 Epoch 11/12
 24/24 - 5s - loss: 0.0131 - accuracy: 1.0000 - val_loss: 0.1792 - val_accuracy: 0.9000
 Epoch 12/12
 24/24 - 5s - loss: 0.0154 - accuracy: 1.0000 - val_loss: 0.2456 - val_accuracy: 0.8667

[[5 0 0 0]
 [0 5 0 0]
 [1 0 4 0]
 [0 0 0 5]]

13 Epochs:

Epoch 1/13
 24/24 - 5s - loss: 1.1442 - accuracy: 0.5208 - val_loss: 0.9838 - val_accuracy: 0.5333
 Epoch 2/13
 24/24 - 5s - loss: 0.2345 - accuracy: 0.9542 - val_loss: 0.7961 - val_accuracy: 0.7000
 Epoch 3/13
 24/24 - 5s - loss: 0.1065 - accuracy: 0.9958 - val_loss: 0.4305 - val_accuracy: 0.8667
 Epoch 4/13
 24/24 - 5s - loss: 0.0748 - accuracy: 0.9958 - val_loss: 0.3506 - val_accuracy: 0.8667
 Epoch 5/13
 24/24 - 5s - loss: 0.0616 - accuracy: 1.0000 - val_loss: 0.3717 - val_accuracy: 0.8667
 Epoch 6/13
 24/24 - 5s - loss: 0.0443 - accuracy: 1.0000 - val_loss: 0.2249 - val_accuracy: 0.9333
 Epoch 7/13
 24/24 - 5s - loss: 0.0308 - accuracy: 0.9958 - val_loss: 0.2019 - val_accuracy: 0.9333
 Epoch 8/13
 24/24 - 5s - loss: 0.0249 - accuracy: 1.0000 - val_loss: 0.2742 - val_accuracy: 0.9000
 Epoch 9/13
 24/24 - 5s - loss: 0.0242 - accuracy: 1.0000 - val_loss: 0.1617 - val_accuracy: 0.9667
 Epoch 10/13
 24/24 - 5s - loss: 0.0149 - accuracy: 1.0000 - val_loss: 0.1310 - val_accuracy: 0.9333

Epoch 11/13

24/24 - 5s - loss: 0.0200 - accuracy: 1.0000 - val_loss: 0.1584 - val_accuracy: 0.9667

Epoch 12/13

24/24 - 5s - loss: 0.0162 - accuracy: 1.0000 - val_loss: 0.2190 - val_accuracy: 0.9333

Epoch 13/13

24/24 - 5s - loss: 0.0089 - accuracy: 1.0000 - val_loss: 0.2896 - val_accuracy: 0.9000

[[3 2 0 0]

[0 5 0 0]

[1 0 4 0]

[0 0 0 5]]

14 Epochs:

Epoch 1/14

24/24 - 6s - loss: 0.9631 - accuracy: 0.6042 - val_loss: 0.8938 - val_accuracy: 0.7000

Epoch 2/14

24/24 - 5s - loss: 0.2278 - accuracy: 0.9583 - val_loss: 0.5337 - val_accuracy: 0.8333

Epoch 3/14

24/24 - 5s - loss: 0.0908 - accuracy: 0.9958 - val_loss: 0.5663 - val_accuracy: 0.7667

Epoch 4/14

24/24 - 5s - loss: 0.0587 - accuracy: 1.0000 - val_loss: 0.4266 - val_accuracy: 0.8000

Epoch 5/14

24/24 - 5s - loss: 0.0517 - accuracy: 1.0000 - val_loss: 0.3542 - val_accuracy: 0.8667

Epoch 6/14

24/24 - 5s - loss: 0.0293 - accuracy: 1.0000 - val_loss: 0.3394 - val_accuracy: 0.8667

Epoch 7/14

24/24 - 5s - loss: 0.0264 - accuracy: 1.0000 - val_loss: 0.2930 - val_accuracy: 0.8667

Epoch 8/14

24/24 - 5s - loss: 0.0216 - accuracy: 1.0000 - val_loss: 0.4282 - val_accuracy: 0.8000

Epoch 9/14

24/24 - 5s - loss: 0.0257 - accuracy: 1.0000 - val_loss: 0.1490 - val_accuracy: 0.9667

Epoch 10/14

24/24 - 5s - loss: 0.0206 - accuracy: 1.0000 - val_loss: 0.3039 - val_accuracy: 0.9000

Epoch 11/14

24/24 - 5s - loss: 0.0162 - accuracy: 1.0000 - val_loss: 0.3657 - val_accuracy: 0.8667

Epoch 12/14

24/24 - 5s - loss: 0.0183 - accuracy: 1.0000 - val_loss: 0.2928 - val_accuracy: 0.8667

Epoch 13/14

24/24 - 5s - loss: 0.0312 - accuracy: 0.9917 - val_loss: 0.3346 - val_accuracy: 0.8667

Epoch 14/14

24/24 - 5s - loss: 0.0082 - accuracy: 1.0000 - val_loss: 0.2047 - val_accuracy: 0.9000

[[4 1 0 0]

[0 5 0 0]

[2 0 3 0]

[0 0 0 5]]

15 Epochs:

Epoch 1/15

24/24 - 5s - loss: 1.1425 - accuracy: 0.5333 - val_loss: 1.0725 - val_accuracy: 0.6000

Epoch 2/15

24/24 - 5s - loss: 0.2714 - accuracy: 0.9250 - val_loss: 0.7950 - val_accuracy: 0.7000

Epoch 3/15

24/24 - 5s - loss: 0.1141 - accuracy: 0.9917 - val_loss: 0.5464 - val_accuracy: 0.8000

Epoch 4/15

24/24 - 5s - loss: 0.0924 - accuracy: 0.9917 - val_loss: 0.4415 - val_accuracy: 0.7667

Epoch 5/15

24/24 - 5s - loss: 0.0504 - accuracy: 0.9958 - val_loss: 0.3289 - val_accuracy: 0.8667

Epoch 6/15

24/24 - 5s - loss: 0.0392 - accuracy: 1.0000 - val_loss: 0.2786 - val_accuracy: 0.8667

Epoch 7/15

24/24 - 5s - loss: 0.0237 - accuracy: 1.0000 - val_loss: 0.2722 - val_accuracy: 0.8667

Epoch 8/15

24/24 - 5s - loss: 0.0315 - accuracy: 1.0000 - val_loss: 0.2478 - val_accuracy: 0.9333

Epoch 9/15

24/24 - 5s - loss: 0.0274 - accuracy: 1.0000 - val_loss: 0.3018 - val_accuracy: 0.9000

Epoch 10/15

24/24 - 5s - loss: 0.0192 - accuracy: 1.0000 - val_loss: 0.1525 - val_accuracy: 0.9667

Epoch 11/15

24/24 - 5s - loss: 0.0168 - accuracy: 1.0000 - val_loss: 0.2568 - val_accuracy: 0.9000

Epoch 12/15

24/24 - 5s - loss: 0.0254 - accuracy: 1.0000 - val_loss: 0.2249 - val_accuracy: 0.8667

Epoch 13/15

24/24 - 5s - loss: 0.0137 - accuracy: 1.0000 - val_loss: 0.1250 - val_accuracy: 0.9667

Epoch 14/15

24/24 - 5s - loss: 0.0109 - accuracy: 1.0000 - val_loss: 0.2319 - val_accuracy: 0.9333

Epoch 15/15

24/24 - 5s - loss: 0.0130 - accuracy: 1.0000 - val_loss: 0.1588 - val_accuracy: 0.9667

[[5 0 0 0]

[0 5 0 0]

[0 0 5 0]

[0 0 0 5]]

Analysis:

We observed that the model correctly classified each car in the test data set after training the model for 15 epochs. The valid accuracy after 15 Epochs was 96.67%, which exceeded our engineering requirement of 92%. We also observed that the valid accuracy was consistently lower than the normal accuracy.

Conclusion:

This specific model was effective for the provided dataset after 15 epochs. However, the values of the valid accuracy were consistently lower than the normal accuracy. This suggested that overfitting was occurring. At this point, the model should be improved by incorporating a larger training dataset containing augmented versions of the original dataset, as well as special cases that would otherwise be misclassified.

Testing with 10 Classes:

Objective:

In this test, we trained the model using 10 classes to observe the effects of a significantly larger dataset on the model's accuracy. This was also meant to observe the scalability of the dashcam in regards to the amount of cars it can accurately classify.

Method:

We trained the model using 60 images of cars per class in the training dataset and 14 unique images of the same car in the validation dataset. We trained the last 20 layers of the model for 100 epochs to try and achieve the greatest accuracy possible, and tested the model against the image of an Acura Integra depicted below.



Fig 1.1 Depicts an image of an Acura Integra used to test the model's ability to classify correctly.

Observations:

Epoch 1/100

24/24 - 12s - loss: 2.0695 - accuracy: 0.2708 - val_loss: 2.1107 - val_accuracy: 0.2667

Epoch 2/100

24/24 - 8s - loss: 1.2244 - accuracy: 0.6208 - val_loss: 1.4169 - val_accuracy: 0.5667

Epoch 3/100

24/24 - 9s - loss: 0.7021 - accuracy: 0.8125 - val_loss: 1.5388 - val_accuracy: 0.5333

Epoch 4/100

24/24 - 9s - loss: 0.4195 - accuracy: 0.9292 - val_loss: 0.9459 - val_accuracy: 0.6667

Epoch 5/100

24/24 - 9s - loss: 0.3779 - accuracy: 0.9333 - val_loss: 1.3045 - val_accuracy: 0.5000

Epoch 6/100

24/24 - 9s - loss: 0.2644 - accuracy: 0.9583 - val_loss: 0.6723 - val_accuracy: 0.8667

Epoch 7/100

24/24 - 8s - loss: 0.1975 - accuracy: 0.9667 - val_loss: 0.6082 - val_accuracy: 0.8667

Epoch 8/100

24/24 - 9s - loss: 0.1633 - accuracy: 0.9917 - val_loss: 0.4870 - val_accuracy: 0.8667

Epoch 9/100

24/24 - 9s - loss: 0.1292 - accuracy: 0.9833 - val_loss: 0.4666 - val_accuracy: 0.8667

Epoch 10/100

24/24 - 9s - loss: 0.1037 - accuracy: 0.9958 - val_loss: 0.3242 - val_accuracy: 0.9333

Epoch 11/100

24/24 - 9s - loss: 0.0872 - accuracy: 1.0000 - val_loss: 0.4225 - val_accuracy: 0.8667

Epoch 12/100

24/24 - 9s - loss: 0.0728 - accuracy: 1.0000 - val_loss: 0.2739 - val_accuracy: 0.9333

Epoch 13/100

24/24 - 9s - loss: 0.0599 - accuracy: 1.0000 - val_loss: 0.2122 - val_accuracy: 0.9333

Epoch 14/100

24/24 - 9s - loss: 0.0560 - accuracy: 1.0000 - val_loss: 0.3068 - val_accuracy: 0.9000

Epoch 15/100

24/24 - 9s - loss: 0.0400 - accuracy: 1.0000 - val_loss: 0.3678 - val_accuracy: 0.8667

Epoch 16/100

24/24 - 9s - loss: 0.0481 - accuracy: 1.0000 - val_loss: 0.2626 - val_accuracy: 0.9000

Epoch 17/100

24/24 - 9s - loss: 0.0457 - accuracy: 1.0000 - val_loss: 0.1305 - val_accuracy: 0.9667

Epoch 18/100

24/24 - 10s - loss: 0.0362 - accuracy: 1.0000 - val_loss: 0.3726 - val_accuracy: 0.8333

Epoch 19/100

24/24 - 9s - loss: 0.0425 - accuracy: 1.0000 - val_loss: 0.1694 - val_accuracy: 0.9667

Epoch 20/100

24/24 - 9s - loss: 0.0385 - accuracy: 0.9958 - val_loss: 0.2648 - val_accuracy: 0.9000

Epoch 21/100

24/24 - 10s - loss: 0.0247 - accuracy: 1.0000 - val_loss: 0.3382 - val_accuracy: 0.9000

Epoch 22/100

24/24 - 9s - loss: 0.0289 - accuracy: 1.0000 - val_loss: 0.1648 - val_accuracy: 0.9667

Epoch 23/100

24/24 - 9s - loss: 0.0381 - accuracy: 1.0000 - val_loss: 0.2546 - val_accuracy: 0.9333

Epoch 24/100

24/24 - 9s - loss: 0.0227 - accuracy: 1.0000 - val_loss: 0.2963 - val_accuracy: 0.9000

Epoch 25/100

24/24 - 8s - loss: 0.0259 - accuracy: 1.0000 - val_loss: 0.1352 - val_accuracy: 0.9667

Epoch 26/100

24/24 - 8s - loss: 0.0185 - accuracy: 1.0000 - val_loss: 0.3356 - val_accuracy: 0.9000

Epoch 27/100

24/24 - 9s - loss: 0.0167 - accuracy: 1.0000 - val_loss: 0.2149 - val_accuracy: 0.9333

Epoch 28/100

24/24 - 10s - loss: 0.0189 - accuracy: 1.0000 - val_loss: 0.2676 - val_accuracy: 0.9000

Epoch 29/100

24/24 - 9s - loss: 0.0171 - accuracy: 1.0000 - val_loss: 0.2253 - val_accuracy: 0.9000

Epoch 30/100

24/24 - 10s - loss: 0.0123 - accuracy: 1.0000 - val_loss: 0.1144 - val_accuracy: 1.0000

Epoch 31/100

24/24 - 9s - loss: 0.0125 - accuracy: 1.0000 - val_loss: 0.1565 - val_accuracy: 0.9667

Epoch 32/100

24/24 - 9s - loss: 0.0147 - accuracy: 1.0000 - val_loss: 0.1737 - val_accuracy: 0.9333

Epoch 33/100

24/24 - 9s - loss: 0.0118 - accuracy: 1.0000 - val_loss: 0.2119 - val_accuracy: 0.9333

Epoch 34/100

24/24 - 9s - loss: 0.0176 - accuracy: 1.0000 - val_loss: 0.3288 - val_accuracy: 0.8667

Epoch 35/100

24/24 - 9s - loss: 0.0126 - accuracy: 1.0000 - val_loss: 0.2006 - val_accuracy: 0.9000

Epoch 36/100

24/24 - 9s - loss: 0.0153 - accuracy: 1.0000 - val_loss: 0.3102 - val_accuracy: 0.8333

Epoch 37/100

24/24 - 9s - loss: 0.0112 - accuracy: 1.0000 - val_loss: 0.1506 - val_accuracy: 0.9333

Epoch 38/100

24/24 - 9s - loss: 0.0100 - accuracy: 1.0000 - val_loss: 0.2611 - val_accuracy: 0.8667

Epoch 39/100

24/24 - 9s - loss: 0.0086 - accuracy: 1.0000 - val_loss: 0.1668 - val_accuracy: 0.9333

Epoch 40/100

24/24 - 9s - loss: 0.0090 - accuracy: 1.0000 - val_loss: 0.2194 - val_accuracy: 0.9333

Epoch 41/100

24/24 - 9s - loss: 0.0166 - accuracy: 1.0000 - val_loss: 0.2258 - val_accuracy: 0.8667

Epoch 42/100

24/24 - 9s - loss: 0.0149 - accuracy: 1.0000 - val_loss: 0.0548 - val_accuracy: 1.0000

Epoch 43/100

24/24 - 9s - loss: 0.0085 - accuracy: 1.0000 - val_loss: 0.1223 - val_accuracy: 0.9333

Epoch 44/100

24/24 - 9s - loss: 0.0066 - accuracy: 1.0000 - val_loss: 0.1294 - val_accuracy: 0.9667

Epoch 45/100

24/24 - 9s - loss: 0.0068 - accuracy: 1.0000 - val_loss: 0.0757 - val_accuracy: 0.9667

Epoch 46/100

24/24 - 9s - loss: 0.0068 - accuracy: 1.0000 - val_loss: 0.1247 - val_accuracy: 0.9333

Epoch 47/100

24/24 - 10s - loss: 0.0046 - accuracy: 1.0000 - val_loss: 0.2017 - val_accuracy: 0.9000

Epoch 48/100

24/24 - 9s - loss: 0.0135 - accuracy: 1.0000 - val_loss: 0.1072 - val_accuracy: 1.0000

Epoch 49/100

24/24 - 10s - loss: 0.0088 - accuracy: 1.0000 - val_loss: 0.1398 - val_accuracy: 0.9667

Epoch 50/100

24/24 - 9s - loss: 0.0083 - accuracy: 1.0000 - val_loss: 0.0929 - val_accuracy: 0.9667

Epoch 51/100

24/24 - 9s - loss: 0.0051 - accuracy: 1.0000 - val_loss: 0.1217 - val_accuracy: 0.9667

Epoch 52/100

24/24 - 9s - loss: 0.0068 - accuracy: 1.0000 - val_loss: 0.1730 - val_accuracy: 0.9000

Epoch 53/100

24/24 - 9s - loss: 0.0068 - accuracy: 1.0000 - val_loss: 0.1433 - val_accuracy: 0.9333

Epoch 54/100

24/24 - 9s - loss: 0.0042 - accuracy: 1.0000 - val_loss: 0.1123 - val_accuracy: 0.9667

Epoch 55/100

24/24 - 9s - loss: 0.0075 - accuracy: 1.0000 - val_loss: 0.0712 - val_accuracy: 1.0000

Epoch 56/100

24/24 - 9s - loss: 0.0054 - accuracy: 1.0000 - val_loss: 0.1906 - val_accuracy: 0.9000

Epoch 57/100

24/24 - 9s - loss: 0.0069 - accuracy: 1.0000 - val_loss: 0.0501 - val_accuracy: 1.0000

Epoch 58/100

24/24 - 9s - loss: 0.0046 - accuracy: 1.0000 - val_loss: 0.2212 - val_accuracy: 0.9000

Epoch 59/100

24/24 - 9s - loss: 0.0062 - accuracy: 1.0000 - val_loss: 0.0603 - val_accuracy: 1.0000

Epoch 60/100

24/24 - 9s - loss: 0.0111 - accuracy: 1.0000 - val_loss: 0.1327 - val_accuracy: 0.9667

Epoch 61/100

24/24 - 9s - loss: 0.0047 - accuracy: 1.0000 - val_loss: 0.0254 - val_accuracy: 1.0000

Epoch 62/100

24/24 - 9s - loss: 0.0048 - accuracy: 1.0000 - val_loss: 0.0277 - val_accuracy: 1.0000

Epoch 63/100

24/24 - 9s - loss: 0.0034 - accuracy: 1.0000 - val_loss: 0.0878 - val_accuracy: 0.9667

Epoch 64/100

24/24 - 8s - loss: 0.0053 - accuracy: 1.0000 - val_loss: 0.0668 - val_accuracy: 1.0000

Epoch 65/100

24/24 - 9s - loss: 0.0052 - accuracy: 1.0000 - val_loss: 0.0898 - val_accuracy: 0.9667

Epoch 66/100

24/24 - 8s - loss: 0.0030 - accuracy: 1.0000 - val_loss: 0.2334 - val_accuracy: 0.9000

Epoch 67/100

24/24 - 8s - loss: 0.0028 - accuracy: 1.0000 - val_loss: 0.1615 - val_accuracy: 0.9667

Epoch 68/100

24/24 - 9s - loss: 0.0035 - accuracy: 1.0000 - val_loss: 0.0891 - val_accuracy: 1.0000

Epoch 69/100

24/24 - 9s - loss: 0.0041 - accuracy: 1.0000 - val_loss: 0.1670 - val_accuracy: 0.9333

Epoch 70/100

24/24 - 8s - loss: 0.0040 - accuracy: 1.0000 - val_loss: 0.2158 - val_accuracy: 0.9333

Epoch 71/100

24/24 - 9s - loss: 0.0031 - accuracy: 1.0000 - val_loss: 0.1672 - val_accuracy: 0.9000

Epoch 72/100

24/24 - 9s - loss: 0.0087 - accuracy: 0.9958 - val_loss: 0.1728 - val_accuracy: 0.9667

Epoch 73/100

24/24 - 8s - loss: 0.0027 - accuracy: 1.0000 - val_loss: 0.1122 - val_accuracy: 0.9667

Epoch 74/100

24/24 - 9s - loss: 0.0040 - accuracy: 1.0000 - val_loss: 0.1205 - val_accuracy: 0.9667

Epoch 75/100

24/24 - 9s - loss: 0.0031 - accuracy: 1.0000 - val_loss: 0.1036 - val_accuracy: 0.9667

Epoch 76/100

24/24 - 9s - loss: 0.0033 - accuracy: 1.0000 - val_loss: 0.0951 - val_accuracy: 1.0000

Epoch 77/100

24/24 - 9s - loss: 0.0041 - accuracy: 1.0000 - val_loss: 0.1554 - val_accuracy: 0.9333

Epoch 78/100

24/24 - 9s - loss: 0.0041 - accuracy: 1.0000 - val_loss: 0.2480 - val_accuracy: 0.9333

Epoch 79/100

24/24 - 8s - loss: 0.0043 - accuracy: 1.0000 - val_loss: 0.2241 - val_accuracy: 0.8667

Epoch 80/100

24/24 - 8s - loss: 0.0020 - accuracy: 1.0000 - val_loss: 0.2097 - val_accuracy: 0.9000

Epoch 81/100

24/24 - 9s - loss: 0.0027 - accuracy: 1.0000 - val_loss: 0.2741 - val_accuracy: 0.8667

Epoch 82/100

24/24 - 9s - loss: 0.0034 - accuracy: 1.0000 - val_loss: 0.1306 - val_accuracy: 0.9333

Epoch 83/100

24/24 - 10s - loss: 0.0039 - accuracy: 1.0000 - val_loss: 0.1057 - val_accuracy: 0.9333

Epoch 84/100

24/24 - 8s - loss: 0.0041 - accuracy: 1.0000 - val_loss: 0.2047 - val_accuracy: 0.9000

Epoch 85/100

24/24 - 8s - loss: 0.0022 - accuracy: 1.0000 - val_loss: 0.0872 - val_accuracy: 0.9667

Epoch 86/100

24/24 - 8s - loss: 0.0040 - accuracy: 1.0000 - val_loss: 0.1095 - val_accuracy: 0.9333

Epoch 87/100

24/24 - 9s - loss: 0.0019 - accuracy: 1.0000 - val_loss: 0.0739 - val_accuracy: 0.9667

Epoch 88/100

24/24 - 9s - loss: 0.0028 - accuracy: 1.0000 - val_loss: 0.0886 - val_accuracy: 0.9667

Epoch 89/100

24/24 - 9s - loss: 0.0020 - accuracy: 1.0000 - val_loss: 0.2087 - val_accuracy: 0.9000

Epoch 90/100

24/24 - 9s - loss: 0.0023 - accuracy: 1.0000 - val_loss: 0.1013 - val_accuracy: 0.9667

Epoch 91/100

24/24 - 9s - loss: 0.0028 - accuracy: 1.0000 - val_loss: 0.1302 - val_accuracy: 0.9333

Epoch 92/100

24/24 - 9s - loss: 0.0020 - accuracy: 1.0000 - val_loss: 0.1249 - val_accuracy: 0.9667

Epoch 93/100

24/24 - 9s - loss: 0.0016 - accuracy: 1.0000 - val_loss: 0.1627 - val_accuracy: 0.9333

Epoch 94/100

24/24 - 9s - loss: 0.0019 - accuracy: 1.0000 - val_loss: 0.1065 - val_accuracy: 0.9667

Epoch 95/100

24/24 - 9s - loss: 0.0017 - accuracy: 1.0000 - val_loss: 0.1099 - val_accuracy: 0.9333

Epoch 96/100

24/24 - 9s - loss: 0.0018 - accuracy: 1.0000 - val_loss: 0.2405 - val_accuracy: 0.8667

Epoch 97/100

24/24 - 9s - loss: 0.0031 - accuracy: 1.0000 - val_loss: 0.1112 - val_accuracy: 0.9333

Epoch 98/100

24/24 - 9s - loss: 0.0019 - accuracy: 1.0000 - val_loss: 0.0316 - val_accuracy: 1.0000

Epoch 99/100

24/24 - 9s - loss: 0.0017 - accuracy: 1.0000 - val_loss: 0.1078 - val_accuracy: 0.9667

Epoch 100/100

24/24 - 9s - loss: 0.0019 - accuracy: 1.0000 - val_loss: 0.1621 - val_accuracy: 0.9000

Class Legend:

{'Acura': 0, 'Aston Martin': 1, 'BMW': 2, 'Dodge': 3, 'Ford SUV': 4, 'Honda Accord': 5, 'Jeep SUV': 6, 'Lincoln Sedan': 7, 'Mercedes-Benz Convertible': 8, 'Volkswagen Beetle': 9}

Confidence:

[[8.6947501e-01 4.4677928e-02 1.3524337e-03 1.5819101e-03 4.4029867e-03
5.1673137e-02 1.6091317e-02 8.8732794e-04 9.5111374e-03 3.4681786e-04]

Matrix Output:

[[1 0 0 0 0 0 0 0 0]
[1 0 0 0 0 0 0 0 0]
[1 0 0 0 0 0 0 0 0]
[1 0 0 0 0 0 0 0 0]

```
[1 0 0 0 0 0 0 0 0 0]
[1 0 0 0 0 0 0 0 0 0]
[1 0 0 0 0 0 0 0 0 0]
[1 0 0 0 0 0 0 0 0 0]
[1 0 0 0 0 0 0 0 0 0]
[1 0 0 0 0 0 0 0 0 0]]
```

Analysis:

Model successfully classified the image as an acura with 86.94% confidence rating. The final matrix output indicates that for each class, the model was able to distinguish the picture of the Acura from the other ten cars that it was trained to learn.

Conclusion:

From this test, we concluded that our model was effective at recognizing a specific car placed into the testing dataset, and demonstrated the accurate-car-recognition functionality that we were striving to achieve for this prototype as per our engineering requirements. We also observed that the model was capable of functioning with a significantly larger dataset than originally tested and that the amount of cars the model can accurately distinguish was scalable. Although these results were favorable to our progress, the model's accuracy diminished below an acceptable level, which warrants further optimization and testing.

Parameter Optimization:

Objective:

In order to achieve an accuracy of at least 92%, further optimization was required. In order to achieve this, it was necessary to test different parametric configurations to produce a more accurate model.

Method:

First, we increased the quality of the training dataset by augmenting the existing images. This was done in order to address the overfitting issue observed in previous tests. Next, we modified the model's training parameters for each trial until acceptable values for the model's valid accuracy and confidence values were obtained. These parameters were as follows: Trained Layers, Steps per Epoch, and Epochs. Finally, for each trial, we tested the model against an image of an Acura Integra.

Observations:

Table VII

Trail #	Trained layers	Steps per Epoch	Epoch	Valid Accuracy	Confidence

1	-20	24	15	90%	69%
2	-20	24	15	90%	68%
3	-20	120	15	96%	85%
4	-20	120	15	93%	95%
5	-20	120	20	93%	91%
6	-20	120	20	96%	92%
7	-36	120	20	97%	83%
8	-36	120	20	100%	76%
9	-36	120	10	100%	88%
10	-15	120	10	93%	71%
11	-20	120	10	86%	97%
12	-20	120	10	96%	63%
13	-20	120	10	86%	74%
14	-20	120	12	96%	84%
15	-20	120	15	96%	84%
16	-20	120	20	100%	91%
17	-20	120	20	96%	95%

Table VII displays the different combination of parameters we used when we tested our neural network, these parameters include the number of layers we retrained, total number of epochs, steps done per epoch, accuracy against the valid dataset, and confidence value of the model

Analysis:

We found that augmenting the existing dataset significantly increased the model's valid accuracy, yielding an average of 94.35% under varying parametric configurations. Also, we found that the trial that yielded the best balance between valid accuracy and confidence rate was trial 17, with values of 96% and 95% respectively.

Conclusion:

At the end of this test, we found that our model had met and exceeded the engineering requirement of which the device must yield an accuracy of at least 92%. Since this was achieved, the model was then ready to be integrated into the final prototype.

Optimizing Runtime:

Objective:

Multiple modifications were needed in order to combat our software's noticeably slow runtime for the final prototype. We suspected a number of elements were contributing to this issue, and in this test we aimed to eliminate said elements.

Method:

We first measured the amount of time it took for multiple functions to complete in order to establish a baseline for improvement. We then tested for speed improvements after supplying more power to the device, modifying the graphical user interface, scaling the model down, and employing a lightweight version of YOLOnet.

Observations:

Initial performance of YOLOnet (computer):

Trial 1: 0.44 seconds

Trial 2: 0.40 seconds

Trial 3: 0.37 seconds

After employing YOLOnet-mini (computer):

Trial 1: 0.43 seconds

Trial 2: 0.43 seconds

Trial 3: 0.40 seconds

Initial GUI load time:

1 minute and 55 seconds

After scaling down GUI:

1 minute and 50 seconds

Initial time 5.0 volt and 2.0 amp power delivery:

Time to boot:

1 minute and 43 seconds

Speed of YOLOnet:

Trial 1: 18.2 seconds

Trial 2: 16.9 seconds

Trial 3: 17.1 seconds

Time to process match:

Trial 1: 1 m 36s

Trial 2: 42s

Trial 3: 15s

Trial 4: 12s

After increasing to 5.0 volt and 3.5 amp power delivery:

Time to boot:

1 minute and 32 seconds

Speed of YOLOnet:

Trial 1: 14.6 seconds

Trial 2: 11.9 seconds

Trial 3: 13.0 seconds

Time to process match:

Trial 1: 1 m 32s

Trial 2: 35s

Trial 3: 14s

Trial 4: 11s

Analysis:

We found that the device's performance increased after supplying more power to the dashcam. We also found that modifying the GUI to a lighter version likewise increased the device's performance. Unfortunately, implementing a lightweight version of YOLOnet had little to no impact on system performance.

Conclusion:

While we did see improvements in the device's performance, the results were rather disappointing. None of the tested modifications were enough to yield a significant amount of speedup, and thus the device's usability remains subpar. More modifications could have been done to improve the runtime speed of the dashcam, such as rewriting the program in C in order to implement the lightweight version of TensorFlow, or swapping the NVIDIA Jetson Nano (2GB) with the 4GB version of itself. However, neither of these options were in our interest of time and cost, thus, the prototype remains as is.

Final Design:

UI:

Our final design changed quite a bit in both hardware and software aspects. When we first began designing our dashcam we intended to have physical buttons that the user will press to navigate the menus of our dashcam. However in 397 when we began building the UI for our dashcam using tkinter in python, we released we could design the on screen buttons to be touch compatible with the LCD screen and we ended up going with a touch screen display.



Fig. 1. Displays the Dashcam UI with the car class buttons and Nvidia Jetson camera feed

Hardware:

When it came to the hardware, we originally wanted to create a standard box shaped design for our dashcam like most other products. However if our product was going to be a touch screen then we had to design the box to be larger to fit the LCD screen so that it would be easier for the user to use the touch screen and not be confined to a small box where they misclick a button. Our original design had the touch screen at the back end of the dashcam facing the driver, at the windshield the camera would be positioned in order to take accurate and clear pictures of the vehicles, and the Nvidia Jetson would go in between them. While this order did match our final design, we ended up placing the dashcam in a reclined position in order to provide more balance and secure it more safely.

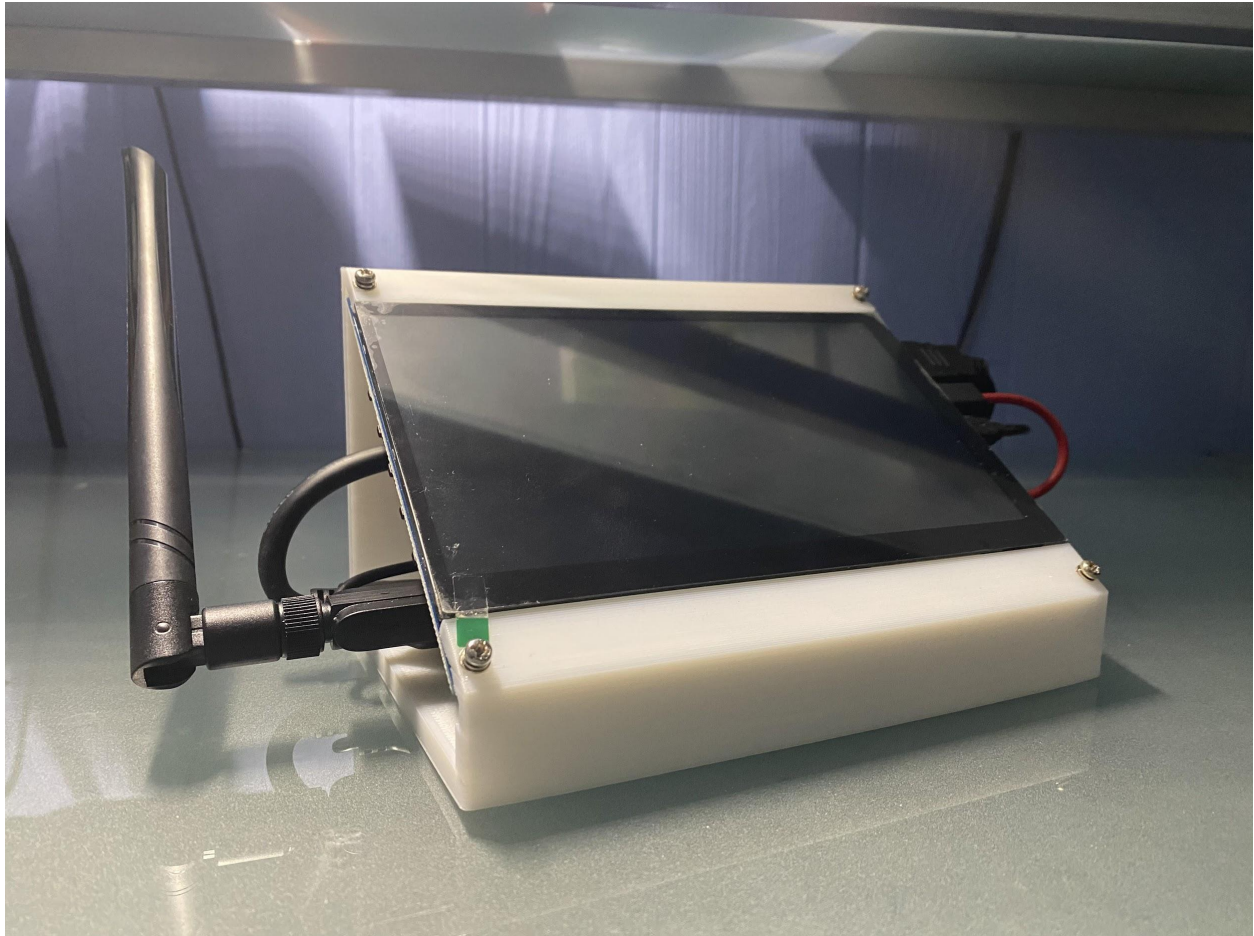


Fig. 2. Depicts an image of the dashcam prototype assembled in its completion.

Image Recognition:

When it came to image recognition, our original 396 design had us reading in images captured by the camera and dividing up the pictures into different sections. These sections would then be compared to images in our library for comparison. For example if we were looking for a Mustang car we would pull up a mustang image from our library and compare each section of the image to see if any car picture taken matches it. We would look at the RGB value to see if the color of the vehicle matches. It would also see if physical features match like say if the mustang horse symbol appeared on both car images then we know that it matches the vehicle model. This was our original 396 design on image identification. For our final design we ended up using transfer learning on neural networks to classify and identify car images

MobileNet:

This semester alongside ECE 397 we also took ECE 491, Introduction to Neural Networks. In this class we learned how to use and train a neural network to recognize images more efficiently. We used transfer learning a neural network MobileNet and retrained it so that it can recognize car images. We trained the mobileNet by giving it access to hundreds of images of cars so that it

can learn to classify a vehicle based on the images it was given. If we wanted to train MobileNet to identify a Honda Civic then we'd feed it dozens of images of different Honda Civics so that it can be trained to recognize that specific car Model. When our microcontroller, the Nvidia Jetson Nano takes a picture of a car, that image is processed by the MobileNet to see if it can classify the image to any of its trained car images.

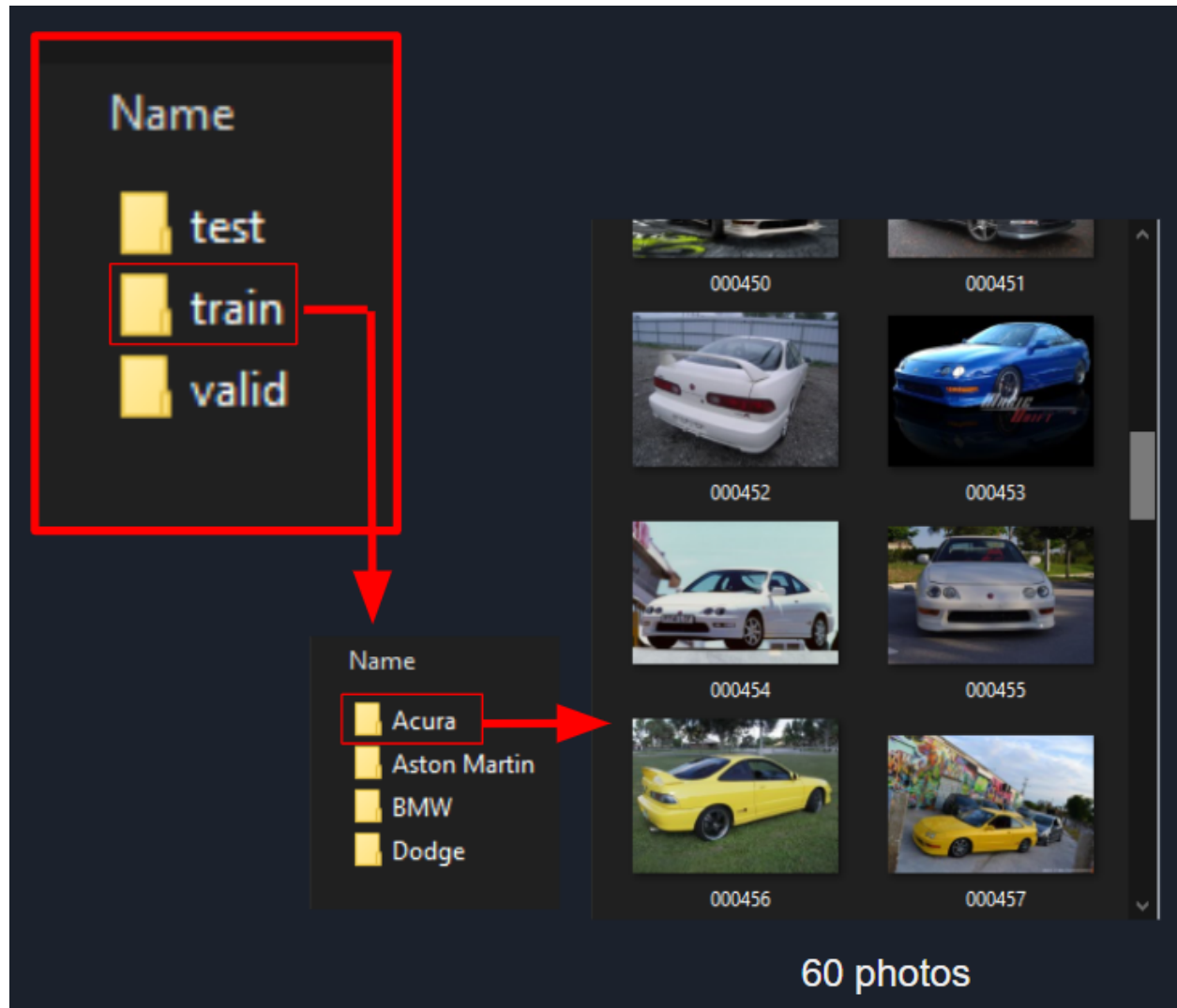


Fig. 3. Displays the directory of training, validation, and testing datasets for training and testing the accuracy of our neural network

YOLONet:

The MobileNet was very useful in processing images captured by the camera but we wanted to make sure our camera only captured images of vehicles and not images of other objects, like trees, buildings, or people. To filter out the camera's live video feed we used the neural network YOLONet. YOLONet is a neural network that can be used to classify objects like MobileNet, but it can also be used on live video feeds. When our program is running, YOLONet would use its classifiers to detect if any object the camera detects is a car or not. If it is, it will save this image

and process it to the MobileNet so that the MobileNet can classify the image to see if it matches our desired vehicle. This saves a lot of resources and time Since we don't have to run every object captured by the camera through our MobileNet neural network



Fig. 4. Depicts object detection performed by YOLOnet to distinguish cars from other objects.

Budget:

The following is a list of components and their prices used in this project.

- Nvidia Jetson Nano 2GB Developer Kit - \$60
- Infrared Night Vision IR Camera Module - \$25
- LCD Screen Bundle - \$60
 - 7 inch HDMI LCD Screen
 - IMX219-77 Camera
 - Micro SD Card 64GB
 - HDMI Cable
 - USB type A plug to micro B plug cable
 - SC Card Reader
 - 15 Pin FFC
 - Jumper Cap
 - Power Adapter (US)
- Lithium Ion Batteries x2 - \$30
- WiFi Module - \$10
- GPS Module - \$10
- **Total: \$195**

Timeline

1/20/21:

- **Eric:** I began researching and watching tutorials on how to connect the nvidia Jetson to its components and how to properly set up the LCD screen.
- **Sai:** I began research on how to code for buttons that will be used in the user interface and also set up Github so my teammates will be always up to date with the codes that are incomplete or completed.
- **Minh:** I researched on how to power the dashcam by searching for the right cable with the right plugs that are used on the Embedded system.
- **Sean:** I searched online for parts that fit the specifications from last semester's final report and submitted the ordering form to the Engineering department. I also researched the dimensions for each component needed and began drafting the first version of the dashcam frame.

1/27/21:

- **Eric:** I began to learn how to program the interface for our dashcam, primarily learning how to use tkinter on Python
- **Sai:** I continue my research further on how to make the button function correctly and the size of the buttons on the user interface and at the same time make sure everyone is connected on Github.
- **Minh:** I collected pictures for a Honda Civic 1998 for a testing for our prototype (around 100 pictures). I also researched the materials for our prototype, 3D printing from our UIC makerspace.
- **Sean:** I created an initial draft block diagram for the overall milestones needed to complete the project.

2/3/21:

- **Eric:** At our meet up at UIC, me and Sean connected our troubleshoot the LCD screen, as well as set up the touch screen functionality
- **Sai:** During this week, I first tested the codes that are compiled to Github on whether or not they are available to my teammates and tested out the wifi router on whether it works with the Nvidia Nano Jetson.

- **Minh:** I compiled the group code to Github so everyone can see it, also I connected the Wifi router to test with our embedded system.
- **Sean:** This week I met up with the team for the first time and began attempting to interface the LCD screen with Eric to make sure it was functional.

2/10/21:

- **Eric:** I successfully created the first iteration of our UI. At this point the UI is able to traverse through several layers of buttons
- **Sai:** I research on the MobileNet functions and check on whether it is compatible with the Nvidia Nano Jetson.
- **Minh:** I continued on working with those images of a car for testing, put them on a single file. I also researched on MobileNet Version 2 to see if this Network fit with our project; uploaded the test code to Github.
- **Sean:** I connected the LCD screen to the NVIDIA Jetson Nano and uploaded Linux Image to get the device to run.

2/17/21:

- **Eric:** I made improvements to the UI such as programming it to scale to the size of whatever display device it's on. Made the buttons larger and added a blue background to make it more visibles.
- **Sean:** I measured all the components for the Dashcam. I also attempted to interface the GPS to the Jetson Nano, but with no success.
- **Minh:** I continued on working with those images of a car for testing, put them on a single file. I also researched on MobileNet Version 2 to see if this Network fit with our project; uploaded the test code to Github.
- **Sai:** I further researched on MobileNet as there were many versions and each of them is improved with each update.

2/24/21:

- **Eric:** I began researching and learning about the different types of neural networks that we could use transfer learning on to retrain to be more accurate with different car models.
- **Sai:** I continue the research on MobileNet coding and see what I will need in order to implement MobileNet into my computer which was helped by Eric.

- **Minh:** I continued doing research on MobileNet Transfer Learning and Installation Guide. I also updated the test code to Github.
- **Sean:** I revised the first draft for the Dashcam frame and updated the dimensions for all the components since the initial dimensions were incorrect. I began learning how to use MobileNet as Professor Cetin advised.

3/3/21:

- **Eric:** I successfully programmed MobileNet to be able to read in images from a directory and accurately classify them. I began and worked in custom class implementations so that it could identify different car models.
- **Sai:** I researched on how to implement classes in the mobilenet to make sure that specific cars can be identified using the MobileNet which was later taken over by Eric as he was the one working on the MobileNet.
- **Minh:** We continued working on the old tasks from last week and updating the codes to avoid flaws before moving to the new tasks.
- **Sean:** I began transferring the draft for the dashcam's frame onto Solidworks, but ran into much difficulty trying to use the program, as it was my first time using the software. I spent the majority of the time researching how to use Solidworks correctly. I also did extensive research on how to use Linux, as well as tested its ability to run Python code.

3/10/21:

- **Eric:** My teacher assistant for ECE 491 helped me find databases for different image libraries that we can use for our project. I programmed a Matlab program to read in the matrix files these images were embedded in and organize them based on their model.
- **Sai:** As the MobileNet was further developed by my teammate, Eric, I started researching and learning about how to connect the camera to Nvidia Nano Jetson and make it work including the camera functions.
- **Minh:** We decided to change from testing images on a 1998 Honda Civic to a 2008 Grey Toyota because it is easier to find pictures for our training dataset. I also downloaded and ran our Mobile Net v2 after 3 weeks of researching it.

- **Sean:** I successfully designed the second version of the dashcam's frame on solidworks. I also tested Eric's UI on the dashcam to check if the LCD's touch functionality was compatible with his program.

3/17/21:

- **Eric:** I began retraining the MobileNet using training, validation and testing folders. These folders would hold images of cars from the database to train the MobileNet and test its accuracy
- **Sai:** I was able to finally implement the code for the camera with its functions for the prototype. It works a bit buggy so I continued to research on how to patch those problems. After completing the code for the camera, the code was then further edited and updated by Sean.
- **Minh:** I began training the module for Mobilenet and added additional class to it. It worked well with Eric's UI.
- **Sean:** I updated the dashcam frame with a slot for the camera component to slide into, as well as hinges to mount the LCD onto. I made some final adjustments to the design and then submitted it to the Makerspace to be built. I also successfully installed all the necessary packages for operating the GPS module, and attempted to interface it with Eric's UI, but with no success. I also figured out how to interface the camera module with the Nvidia jetson. I also collaborated with Eric on the Mobilenet, and figured out how to train the model to accurately classify cars in a confusion matrix.

3/24/21:

- **Eric:** Once the MobileNet was successfully trained, I began creating multiple classes that we can use for the project. I ended up using 10 different car classes
- **Sai:** I started my research on how to implement and code Yolonet. I research different websites to see which methods would be the most suitable for our project.
- **Minh:** I started researching on Yolonet, and figured out how the network functions from different resources.
- **Sean:** I worked with Eric to figure out how to configure the neural network to become more accurate. I learned how to change certain parameters to adjust the

model's accuracy, and also installed NVIDIA CUDA libraries onto my computer so that it could use my GPU to train the model faster.

3/31/21:

- **Eric:** While the MobileNet was accurate in classifying cars that it was trained with, if it was given a car image that it hasn't been trained with then it would classify that car as the closest image based on its dataset. To fix this issue I extracted a confidence value from the trained neural network model that would specify percentage wise how confident it was in its classification. If the confidence was below a certain level then it would classify the image as a "No Match"
- **Sai:** I finally completed coding for the code for the Yolonet. The Yolonet works fine and was able to accurately point out what each part of the pictures are with how many percent that it is sure about each specific object. At the same time, squares are formed around each part of the picture so that the user will be able to see what the objects in the pictures are.
- **Minh:** I worked on coding for the Yolonet after research from last week. The network was able to detect objects on the street such as vehicles, pedestrians or traffic lights and provide a confidence rate of the detection (ex: 0.88/1). I also made all the pictures from the camera processed through Yolonet saved in a particular folder with different names.
- **Sean:** I picked up the dashcam frame from the Makerspace and began assembling the parts together. Although all the parts fit, there were some impracticalities with how some of the components were positioned, and so I made some final revisions to the design on Solidworks to include a slot for the Jetson Nano to fit into as well as a better way to mount the camera so that it would be higher up on the frame and could be angled in a customizable manner in order for the dashcam to be able to see its field of vision on multiple car dashboards.

4/7/21:

- **Eric:** I reworked the AI so that it had one central menu that could be accessed very easily. Me and Sean worked tirelessly to troubleshoot and have the program running on the Nvidia Jetson Nano and install all the necessary libraries.

- **Sai:** I review through the project report done in ECE 396 and check to see if the project meets the objectives and engineering requirements.
- **Minh:** I reviewed and updated all the tasks and made sure the report meets the requirements.
- **Sean:** Eric and I worked on merging the user interface with the mobilenet and camera functionality. This was arguably the most difficult part of the project thus far. We tried different ways of implementing the camera window into the UI, as well as tried out different versions of how the user should interact with the program. We ran into non stop issues trying to fix how the user interface would loop without interfering with the live feed of the camera. We also modified the button functionality to connect to Sai's YOLOnet and the mobilenet. I built the file handling functionality of the UI so that it would place a captured image into the correct file path to be processed by the YOLOnet and then by the mobilenet.

4/14/21:

- **Eric:** The project was completed but I wanted to make it run faster so I made a new function that trained the neural network on a desktop and then exported the trained model. Our main program would then import the model on the Nvidia Jetson Nano
- **Sai:** I checked the tasks that needed to be done before the EXPO to make sure that we do not miss anything that will make our project incomplete.
- **Minh:** I revised all the tasks and was ready for our EXPO live Q&A to make sure nothing of the group is missing .
- **Sean:** I transferred the nearly finished program from the computer onto the final prototype, and troubleshooted any issues the dashcam was having trying to run the program. I modified the final UI, camera code, YOLOnet code, and mobile net to be compatible with the dashcam and installed all the necessary packages to get the program to run successfully. I incorporated the dashcam's GPS functionality into the UI, to finally synthesize the final prototype. I ran some tests and attempted to make the dashcam execute the final program faster, but my efforts were not enough to make any significant improvements to the program's runtime.

4/21/21:

- **Eric:** All necessary updates were completed so I just practiced and rehearsed for the EXPO
- **Sai:** The project was finally completed with no tasks missed out and I started preparing and practicing for the EXPO.
- **Minh:** Everything was ready for EXPO.
- **Sean:** We finished up our presentation and prepared for the EXPO Q&A.

Contributions

Eric: Overall I think I contributed an enormous amount to this project. I worked on the UI, MobileNet, and the integration of all of the code for the project. In the first half of the semester I was the first person with any working code. During our meeting with Professor Revlo and our mentor Professor Cetin I was the only one that showed what I accomplished, the UI and the MobileNet. I consistently reached out to Professor Cetin and his TA Hongyi for advice on how I should approach certain aspects of our code. I downloaded and categorized over 16,000 images of cars for our neural network. Without which we would've had to manually download hundreds of images for the neural network. I learned how to use tkinter and taught my team members how to use tkinter for the UI. The first two iterations of the UI were developed by me and the third iteration Sean and I worked together to get the video feed working. I did extensive amounts of research on how to use MobileNet while coding the UI and taught my teammates how to use and retrain the MobileNet. The training functions were developed by me and I trained the MobileNet using my large car database. Sean and I worked together to improve the accuracy of our model and we both worked on the testing document. Once Sai finished the YOLOnet program, I had to update and make changes to it so that it could be integrated with our other programs. I designed the flowthrough of our project: the UI would boot up and allow the user to select a car, YoloNet would evaluate the objects in the camera and take a picture if it detected a car, MobileNet would then classify the car to see if it matched, if it did then the User would be notified with an onscreen message. I extracted the confidence value from MobileNet to prevent any false positives from occurring on our dashcam. I created a function that trains the model on a desktop, exports it, and then is imported by the Nvidia Jetson Nano. I've spent many sleepless nights towards the end of March and beginning of April trying to ensure the UI and MobileNet functioned correctly with the rest of the project. This is why I think I contributed an enormous amount to this project.

Sai: Summing up the amount of work I did for the project, I believed that I contributed greatly to the project. I researched on the MobileNet functions, set up the Github repository, worked on the Camera UI with complete functions in it, researched on how the Yolonet works, code the Yolonet and connect the Yolonet to the camera code. First when I started the project with my teammates,

I made sure the GitHub repository was set up so that everyone in my team will be able to fully upload and access the codes that will be done during the process of completing the prototype of our project. Then, I started the research on the MobileNet functions and checked to see if it is compatible with the Nvidia Nano Jetson. Then, I studied how to implement the MobileNet into my computer which was helped by Eric as he was the one able to get the MobileNet running on the computer. I then started the research on how to implement the UI with the help of Eric as he had already researched and worked on it for the project. Later, I started working on the camera to be able to work with our dashcam so I started working on its UI and functions which were finally completed with some errors and bugs. As Sean was the one who had the dashcam, the camera codes were further tested out and improved by him. I then continued the research on how to implement classes for the MobileNet but as the main code for the MobileNet was coded and researched by Eric, he took over the research and continued working on it. Finally, I researched on the YoloNet and started working on the coding part for it. I was able to import the files and weight needed for the YoloNet to accurately figure out each object in the picture. Then, I make sure that squares with different colors are formed in the picture with the types and percent shown on the top border of the square to make sure the user is able to see the objects identified. After the completion of YoloNet, the codes are being uploaded to GitHub which was then later connected to the main UI code with some changes by Sean and Eric. Overall, I believed that I contributed greatly and completed my tasks assigned to me in this project.

Minh: Throughout 2 semesters with this project, beside doing researches, I contributed to a good amount of works and the main contributions were functioning MobileNet, powering the prototype, setting up and uploading codes to Github, collecting training dataset for our prototype testing, coding for YoloNet and connect it with the camera of teammate's work. Details of work are mentioned in the weekly tasks above. We had difficulties as well as easiness but mostly were hard times and we all went through it, and I could feel that the results we got were worth the time we spent. I preferred meeting up and working together, especially those Solidworks that we connect our parts together but because of this Covid-19 pandemic, meeting up was not a good idea. Overall, this dashcam project was a success of a whole group and I am proud to be a part of this win.

Sean: I feel as though I made a significantly large contribution to this project. I worked ridiculously hard to push this project to completion, and was really determined to win first place in our category. During the first weeks of the semester, the direction of the project was somewhat unorganized, and a lot of time was spent trying to learn how to use different programs, python libraries, and Linux functions. However, as EXPO got nearer and nearer, I feel like I was ready to implement what I learned for the project. I started by interfacing the components with the dashcam to try to get all the parts ready for integration into the final prototype. Because of COVID-19, meeting up with the group was not ideal, and since I had some experience with Linux, the dashcam was slowly assembled on my desk throughout the whole semester. Because I had the parts with me, I ended up having to interface all the components to the Jetson Nano myself. Figuring out how to make the camera work with the Jetson Nano wasn't that hard, but I had a significantly hard time getting the GPS to work, let alone extract the data retrieved from the GPS and import it into Python. Because I had all the parts with me, it only made sense for me to also design the dashcam frame. I had to buy extra tools and hardware out of my own pocket to make sure that I was making quality measurements and so that the Dashcam would be built correctly on Solidworks without having to have to keep sending drafts to the Makerspace. I also had to buy a faster microSD card and car outlet adapter with a larger power rating in order to try to speed up the program's runtime. Eric and I worked really hard together throughout the semester to get the Mobilenet and UI working correctly, and if it weren't for our hard work at the end of the semester, the project would have failed. I made a lot of contributions to Eric's Mobilenet, optimizing the accuracy, manipulating the datasets, conducting many tests, reworking the final UI, coding all the file handling, integrating the camera functionality into a window in Tkinter, debugging issues with the Tkinter loop, synthesising all the components together, tying all the Python files together, and modifying it all to work on the Dashcam/Linux. Although I am disappointed with how the project did not reach its full potential, I am glad that we were able to finish it in time.

Lessons Learned

Eric:

Throughout this project I learned that it is important to have a structured plan and research on how to approach the software aspect of our project. When I first started working with the neural network I wasn't sure which one I should. I tried learning how to use MobileNet, YoloNet, ResNet, and VGG16. I spent a lot of time focusing on which one was easier to use and train to make the project easier. When in the end I ended up wasting a lot of time learning how to use different neural networks instead of focusing on one and mastering it. At one point I was trying to write the program in Matlab only to find out that our Nvidia Jetson Nano does not work with Matlab. Had I been better prepared and done more research on the Neural Networks and our micro controller I could have saved a lot of time and headaches trying to work with different neural networks

Sai:

After working and finally completing this project, I was able to obtain many knowledge and skills that help me learn and improve in many ways. One of the main things that I learned would be to always communicate with your teammates and plan out on how to tackle problems faced during the completion of the prototype of our project. Another one would be the coding part which is the understanding of how to write codes for the user interface and how to connect the interface with the camera and code camera functions in it. I learnt how to use tkinter for designing the user interface where I learnt how to create buttons, add image in it, edit the shape,size,position and fonts of the letter in the button, and which also included the background colors and the size of the user interface. Finally, the most knowledgeable skills that I learnt is about Yolonet which is part of machine learning. It is an interesting coding skill that I considered hard to learn but really helpful and interesting. I have always been interested in how people coded this Yolonet as it requires the computer to first scan for an image or a video feedback and create square shapes brackets around each specific object while also identifying and calculating the percentage of what are the specific objects in the picture. Overall, I am really happy that I was able to expand my knowledge on AI detection and the amount of time spent on this project was gold as it is really helpful for my future career.

Anh Minh:

After spending 2 semesters with this project, I have learned so many things that helped me improve myself. Besides all those educational and engineering skills, one of the most important ones is teamwork. Without teammates, I wouldn't be able to finish this project successfully. I have to say coding was the most difficult part and took most of the time we worked together and off-meeting but we are happy with the results we received. As if only we could start over, we would spend more time on researching the Networks, pick one and master it instead of trying everything that wasted lots of time. 3D printing, Solidworks, combining all the parts with the codes are something new that I haven't done before, which made the project become even more enjoyable. I preferred meeting up and working together, especially when we combined our parts for the prototype, but because of this Covid-19 pandemic, meeting up was not a good idea. Overall, the time we spent was worth the knowledge we received.

Sean:

I believe that I gained quite a significant amount of experience and knowledge from this project. Nearly everything I accomplished from this project was learned as I progressed, from learning Solidworks, creating my first 3D print, understanding how machine learning works, using Linux based software, building a complex graphical user interface, to integrating it all onto an embedded system. Not too long ago, struggling through college for the first few years with the wrong mindset and the wrong priorities painted a really poor inner image about my confidence in becoming an Engineer. For the first half of my six years in the college of engineering, I've contemplated whether or not it was even worth being in this field of study. Aside from gaining some confidence in my engineering skills towards the end of my time at UIC, this project really helped me to better see the beauty of engineering and my place in the field. I learned how it feels like to have an engineering vision come to life, especially one that did not seem so easily attainable. I learned what it means to collaborate with other people who work hard to bring a project into reality, and I also learned what it means to work with those who don't. I see this project as valuable life experience that I will carry with me when I get a job in the field, and I am really pleased to have used the engineering mindset that UIC has given me to work on a project that matters. I am grateful to senior design for giving me an opportunity to use the skills I've learned to build something that has the potential to make a difference.

Conclusion

The AI-dashcam was finally completed which met almost all of the objectives that we planned out. It scanned the car that is specifically told to be searched using the MobileNet and the Yolonet. First, it processes the video feed of the Nvidia Jetson camera. If a car is detected then it will crop out other objects in the camera and take a picture. The image is then processed to MobileNet which identifies whether the car that is scanned matches the suspected car. But there are some improvements that can be done. One of the improvements would be the number of car models that can be identified. Due to the limited amount of time and the specification of the processor used was not powerful enough, only 10 types and models of cars can only be identified. Another improvement that can be done would be the user interface. Currently the user interface is simple with just 10 buttons representing the 10 different cars. We would be able to add a main UI where the user will be able to choose to use the normal function camera and the specific AI which will also allow the user to choose between the video feedback of the camera with yolonet or just the camera with the AI working behind the scene to be shown.

References

- [1] G. Falcon, Anderson Cooper Blog, CNN, January 15, 2007. Accessed on: March 1, 2021. [Online]. Available:
<http://www.cnn.com/CNN/Programs/anderson.cooper.360/blog/2007/01/raw-data-kidnapping-statistics.html>

- [2] A. Thompkins, “Thursday Edition Kidnapping Facts”, Poynter, 2003, Accessed on March 15, 2021. [Online]. Available:
<https://www.poynter.org/reporting-editing/2003/thursday-edition-kidnapping-facts/>

- [3] Kim, J. “*Neural Network Learning Method and Device*”, US 20200285965A1, Sept. 10, 2020

- [4] Jjie Wang and Zihao Li 2018 IOP Conf. Ser.: Earth Environ. Sci. 170 032110

- [5] Taigman, Yaniv, et al. “DeepFace: Closing the Gap to Human-Level Performance in Face Verification.” *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, doi:10.1109/cvpr.2014.220.

- [6] J. Huang, *Jetson Nano 2GB Developer Kit User*, NVIDIA, October 5, 2020. Accessed on February 1, 2020. [Online]. Available:
<https://developer.nvidia.com/embedded/learn/jetson-nano-2gb-devkit-user-guide>

- [7] Li, Fei-Fei, Computer Vision, Topic: “Image Classification” CS231n, Stanford University, Stanford California, Aug 11, 2017.

Appendix A

Survey Results:

<https://docs.google.com/forms/d/1fPSXyHimNFRIXFCmXYgtTUutxF4lDo4N-cGKnYykU-k/edit#responses>

Survey Image A:

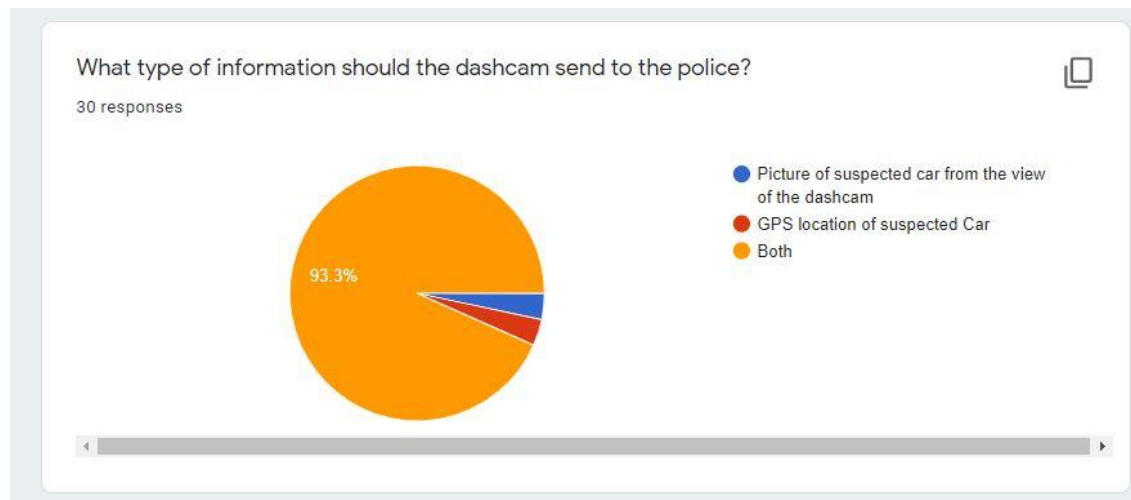


Fig. 1. Displays a pie chart of what type of information surveyed users would want to see in our dashcam

Survey Image B:

Should other information be sent to the police? If so, what information?

19 responses

- Idea: this could help police find stolen cars as well. I think that a simple alert might be useful as well.
- License Plate number
- no
- Date and Time
- Possible the color or type of the car, if AI can differentiate between a sedan or a minivan etc
- time
- place
- driver gender , race, hair, and any other descriptions
- What city, and state the AI picked up as well as time the car was seen.
- Color of the car, license plate number
- ..

Fig. 2. Lists several pieces of information that can be sent to the police that our surveyed users suggested

Survey Image C:

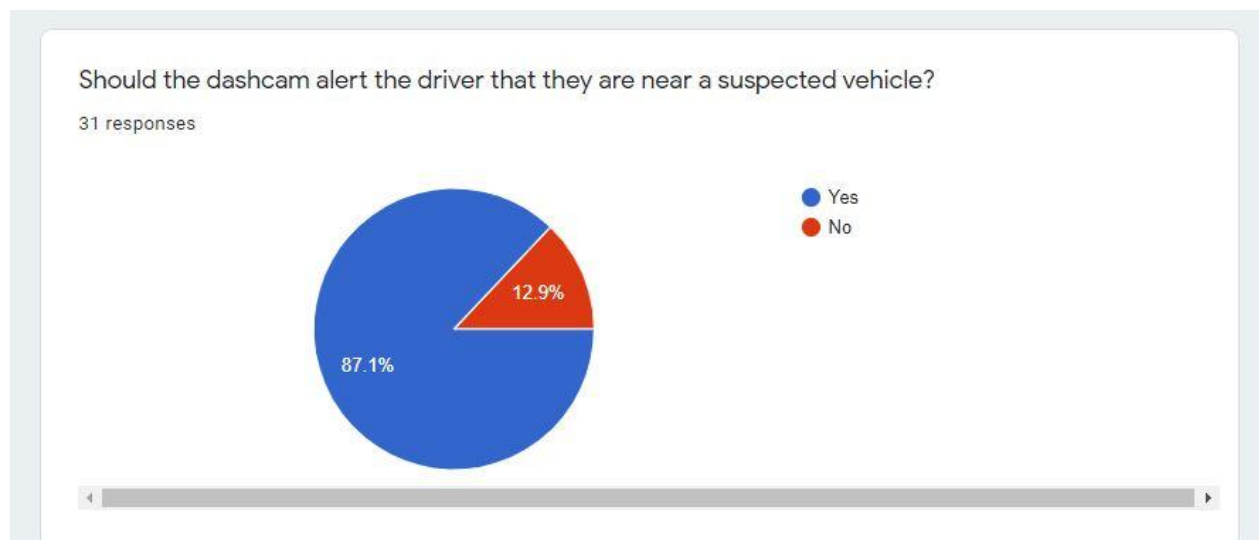


Fig. 3. Displays a pie chart with user responses on whether or not the dashcam should alert the driver that they are near a suspected vehicle

Survey image D:

Do you feel that having a dashcam can potentially report this information to the police, a violation of privacy? If yes, explain why. If no, leave blank.

25 responses

No

No, because this information is being used to help someone that might need help in being found. As long as the information is being used in that sense.

I don't think it will be a violation as long as it follows these guidelines:

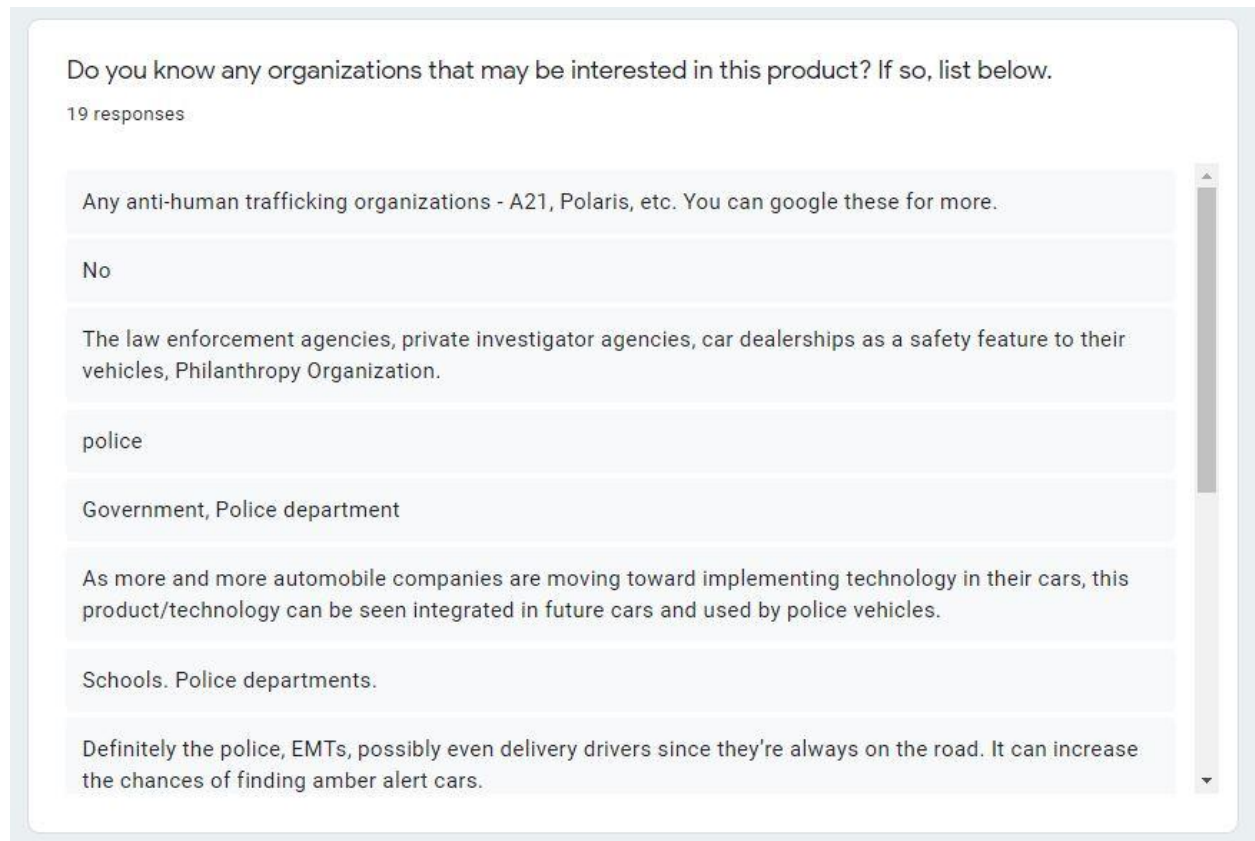
- There is a warning on the product of what it entails, so that consumers are aware of what the dashcam will do, i.e. if it records audio, make sure this is labeled. If it takes pictures/videos, make sure this is labeled as well.
- Maybe make a signed waiver/contract or something for the previous issue bullet point
-

No, we are not allowing the criminal get away with a crime. This is not a private matter when someone's life is in danger.

no

Not at all it's for the safety

Fig. 4. Shows a list of responses by our surveyed users when asked if our product could be considered a violation of privacy

Survey Image E:

Do you know any organizations that may be interested in this product? If so, list below.

19 responses

Any anti-human trafficking organizations - A21, Polaris, etc. You can google these for more.

No

The law enforcement agencies, private investigator agencies, car dealerships as a safety feature to their vehicles, Philanthropy Organization.

police

Government, Police department

As more and more automobile companies are moving toward implementing technology in their cars, this product/technology can be seen integrated in future cars and used by police vehicles.

Schools. Police departments.

Definitely the police, EMTs, possibly even delivery drivers since they're always on the road. It can increase the chances of finding amber alert cars.

Fig. 5. Displays our users' response when asked what other organizations would be interested in our product

Survey Image F:

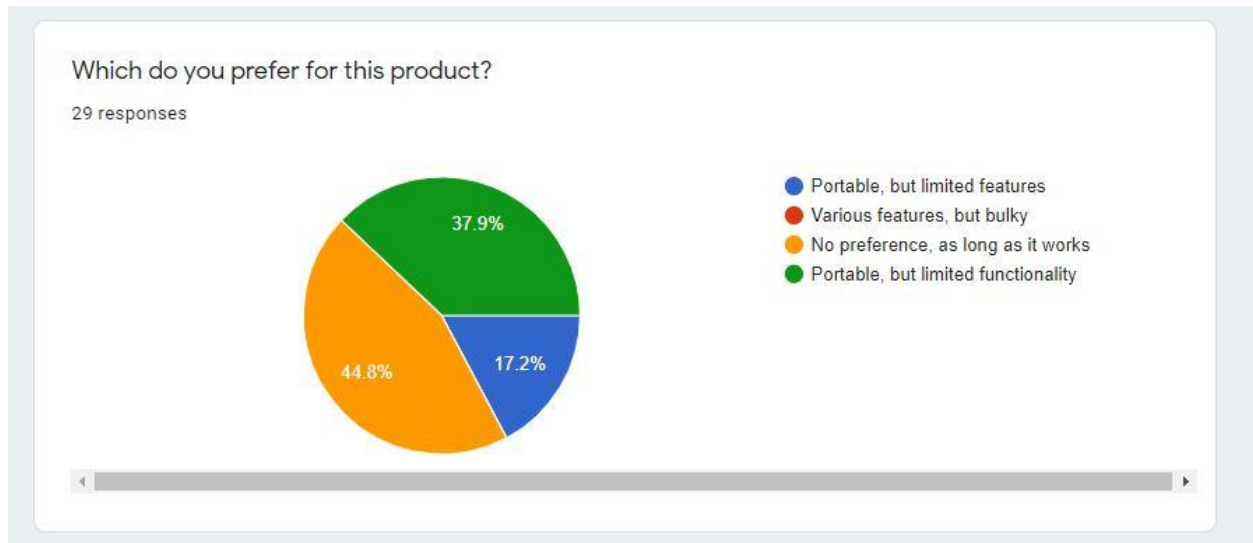


Fig.6. Shows a pie chart where users were asked what preference they had in regards to our products size and number of features

Survey Image G:

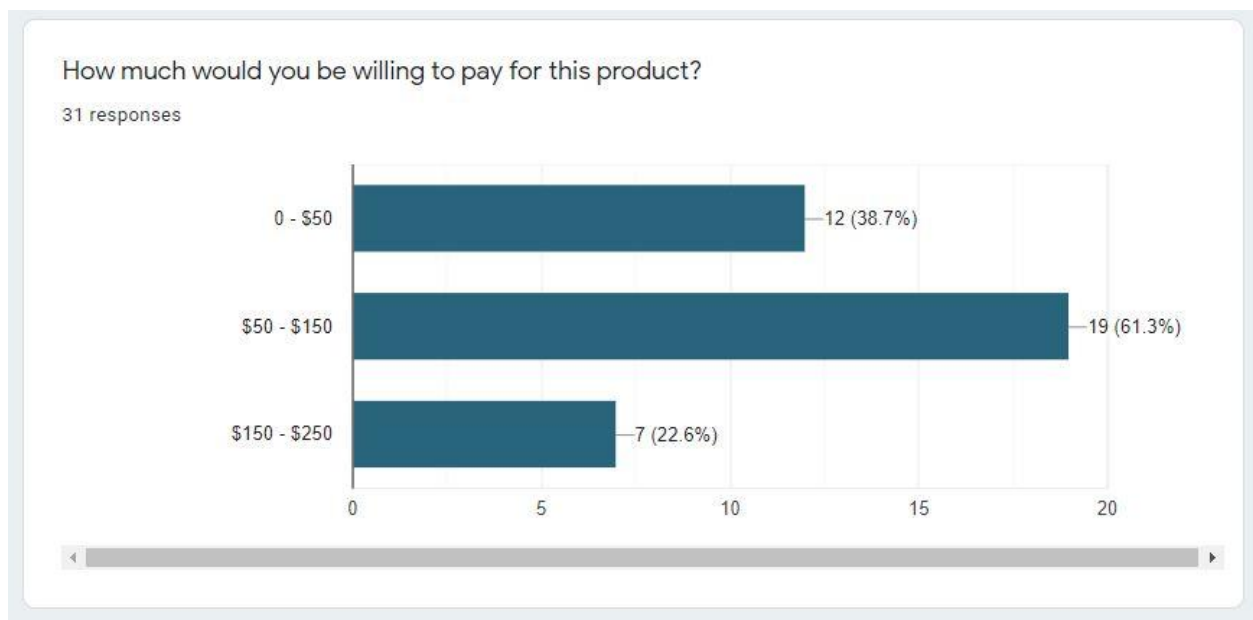


Fig. 7. Displays a bar graph where surveyed users voted on what price they would be willing to pay for the dashcam

Survey Image H:

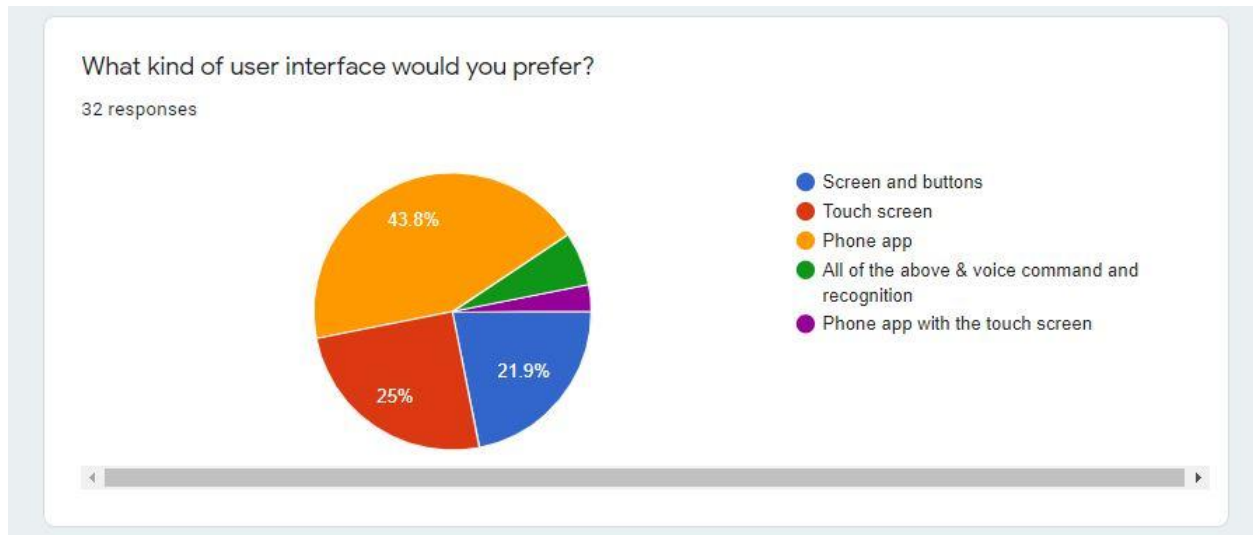


Fig. 8. Shows a pie chart where users voted on what type of interface they'd prefer

Survey Image I:

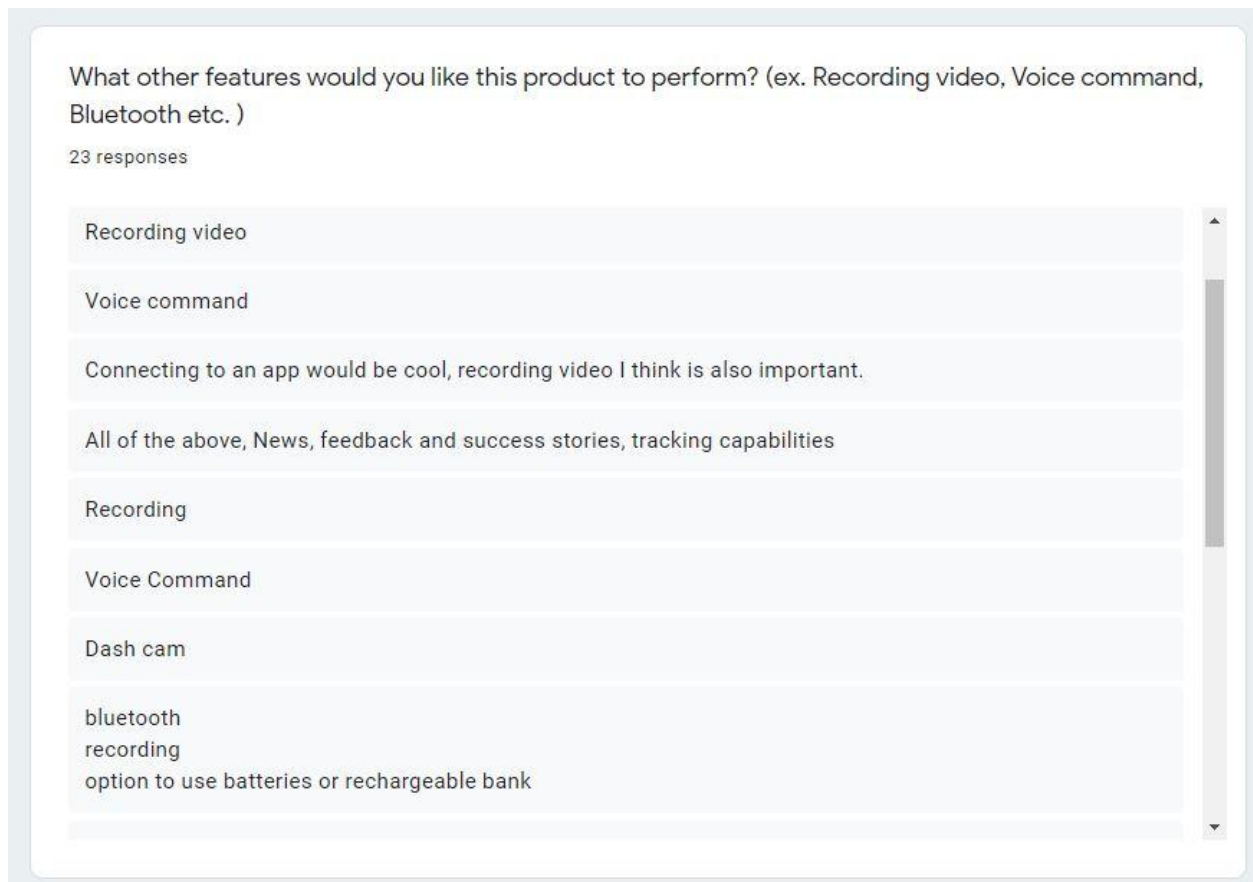


Fig. 9. Shows user responses when asked what other features they want our product to perform

Survey Image J:

Any possible concerns you might have regarding this product?

18 responses

That the information or data could be sold to a third party.

I would keep in mind that when you are creating this product, that it is important to keep in mind of the accessibility it offers. Like, if there is a phone app, I would make the phone app an optional thing for the product and not the primary use of it because not everybody will have a cell phone or that you're not supposed to be texting and driving.

Also, I would make it simple enough that when you make this device, that it is easy to use so that the person can just press one button and not be distracted or have to put their focus elsewhere when driving (this could create accidents).

Accuracy in identification to prevent unnecessary implications of the innocent public.

no

Fig. 10. Lists concerns raised by users on our product

Appendix B

Initial concept Design 1:

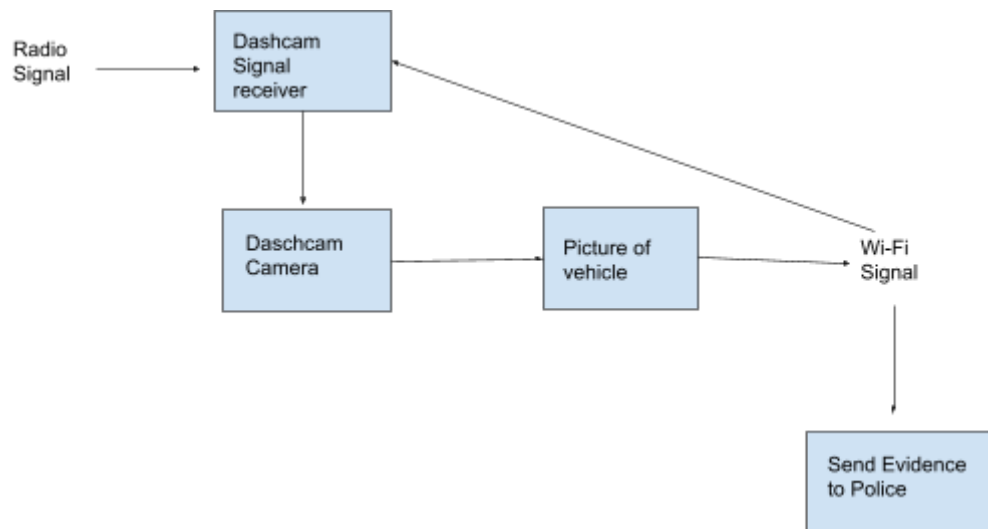


Fig. 1. Shows our initial design of our project flowthrough on how all the components should interact and in what order.

Initial Concept Design 2:

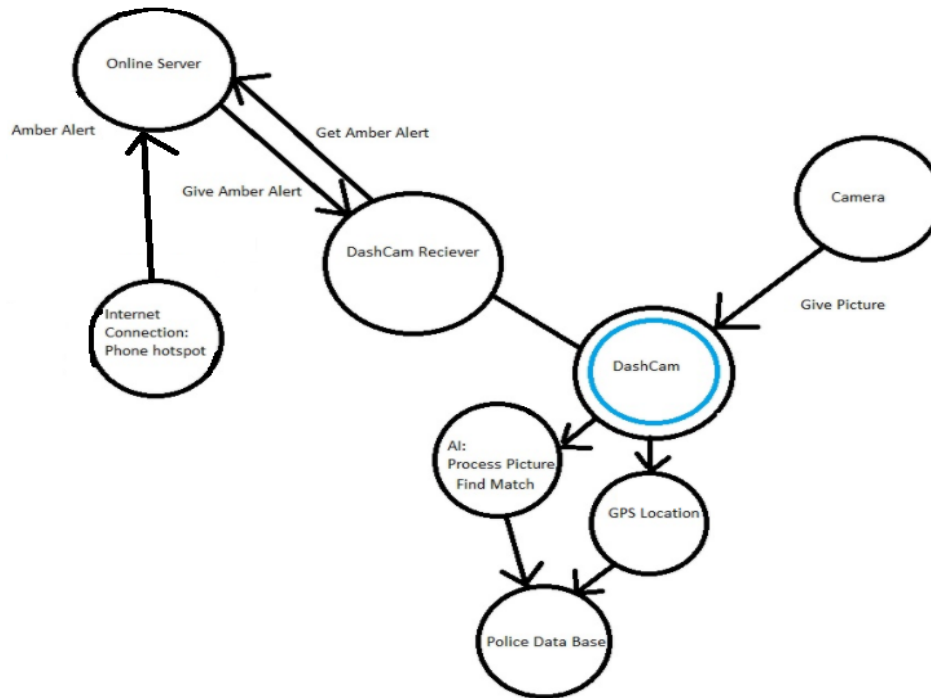


Fig. 2. Depicts an early concept for the handling of software and hardware functions.

Initial concept Design 3:

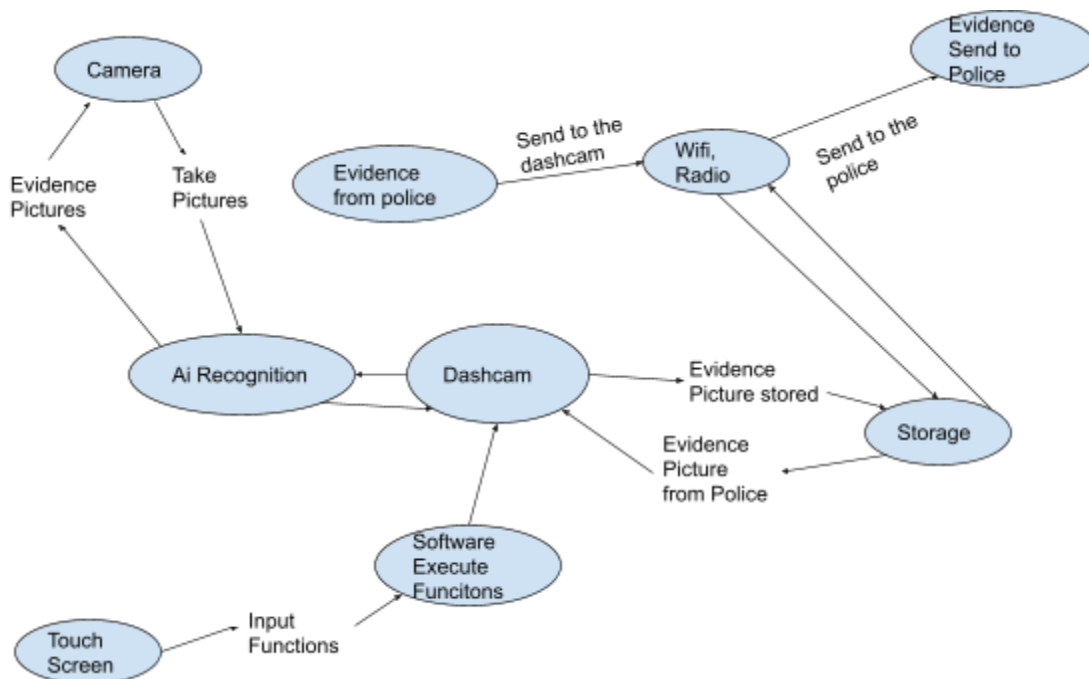


Fig. 3. Depicts an early concept flow chart of subfunctions and hardware.

Initial concept Design 4:

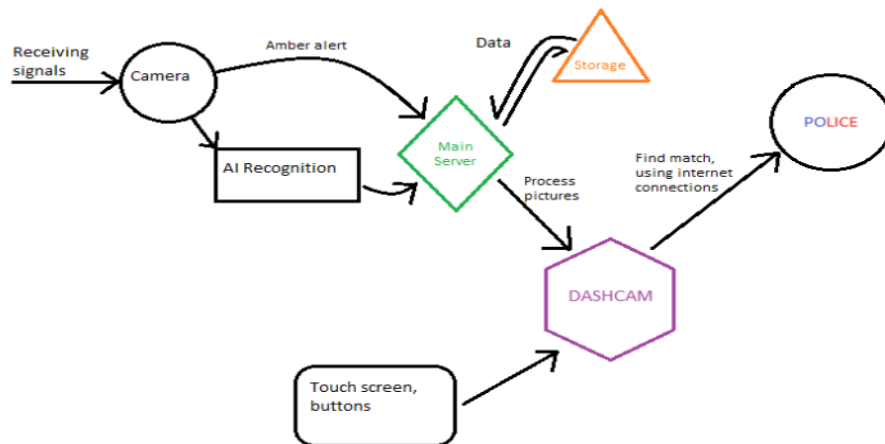


Fig. 4. Depicts an early concept flow chart of subfunctions and hardware.

Appendix C

Eric's Pairwise comparison of the selection criteria:

Table VIII

	Criteria 1	Criteria 2	Criteria 3	Criteria 4	Criteria 5	Geometric Means	Weights
Criteria 1	1	3	5	7	9	3.93	0.511
Criteria 2	1/3	1	3	5	7	2.03	0.26
Criteria 3	1/5	1/3	1	3	5	0.99	0.128
Criteria 4	1/7	1/5	1/3	1	1/3	0.48	0.0625
Criteria 5	1/9	1/7	1/5	1/3	1	0.25	0.032
						7.68	1

Table VIII displays Eric's ranking for the pairwise comparisons on which ones are more important than the other

Sai's Pairwise comparison of the selection criteria:

Table IX

	Criteria 1	Criteria 2	Criteria 3	Criteria 4	Criteria 5	Geometric Means	Weights
Criteria 1	1	5	7	7	9	4.66	0.55
Criteria 2	1/5	1	5	7	7	2.18	0.26
Criteria 3	1/7	1/5	1	9	5	1.05	0.12
Criteria 4	1/7	1/7	1/9	1	1/3	0.24	0.03
Criteria 5	1/9	1/7	1/5	3	1	0.39	0.05
						8.52	1

Table IX displays Sai's ranking for the pairwise comparisons on which ones are more important than the other

Minh's Pairwise comparison of the selection criteria:

	Criteria 1	Criteria 2	Criteria 3	Criteria 4	Criteria 5	Geometric Means	Weights
Criteria 1	1	1	5	3	7	2.54	0.38
Criteria 2	1	1	5	3	7	2.54	0.38
Criteria 3	1/5	1/5	1	1	1	0.53	0.08
Criteria 4	1/3	1/3	1	1	1	0.64	0.1
Criteria 5	1/7	1/7	1	1	1	0.46	0.07
						6.71	1.00

Sean's Pairwise comparison of the selection criteria:

Table X

	Criteria 1	Criteria 2	Criteria 3	Criteria 4	Criteria 5	Geometric Means	Weights
Criteria 1	1	7	5	3	3	2.46	0.40
Criteria 2	1/7	1	3	5	5	1.60	0.26
Criteria 3	1/5	1/3	1	1/3	5	0.64	0.10
Criteria 4	1/3	1/5	3	1	7	1.06	0.18
Criteria 5	1/3	1/5	1/5	1/7	1	0.28	0.05

Table X displays Sean's ranking for the pairwise comparisons on which ones are more important than the other

Table of Identification and Rate of Alternatives Relative to the Criteria

Table XI

	Criteria 1	Criteria 2	Criteria 3	Criteria 4	Criteria 5
Alternative 1	5	5	3	3	5
Alternative 2	3	5	5	5	3
Alternative 3	3	5	5	1	3

Table XI rates how well each design alternative satisfies each criteria

Team Weekly Task Document for Spring 2021

- Week 2, 1/2/2021

Task #	Description	Assigned To
1	Order Hardware parts for the project	Sean Atangan
2	Working on how to power the dashcam	Anh Minh Ngo
3	Working on interfacing the LCD screen with the Jetson	Eric Escobedo
4	Working on button functionality	Sai Arkar Khant
5	Design the Dashcam frame	Sean Atangan

6	Set up GitHub	Sai Arkar Khant
---	---------------	-----------------

- Week 3, 1/27/2021

Task #	Description	Assigned To
1	Set up meetings with Cetin	Eric Escobedo
2	Research Coding the buttons	Sai Arkar Khant
3	Research TensorFlow	Sean Atangan
4	Connect everyone to GitHub	Sai Arkar Khant
5	Collect Data Base of Honda Civic 1998	Anh Minh Ngo
6	Build Block Diagram for overall system	Sean Atangan
7	Research Coding for Interface	Eric Escobedo
8	Research Materials for Prototype(makerspace)	Anh Minh Ngo

- Week 4, 2/3/2021

Task #	Description	Assigned To
1	Nvidia Jetson LCD Screen connection	Eric Escobedo
2	Nvidia Jetson LCD Screen connection	Sean Atangan
3	Compile group code to github	Sai Arkar Khant
4	Compile group code to github	Anh Minh Ngo
5	Connect Wi-Fi to Nvidia	Sai Arkar Khant
6	Connect Wi-Fi to Nvidia	Anh Minh Ngo
7	Connect GPS to Nvidia	Eric Escobedo
8	Connect GPS to Nvidia	Sean Atangan

- Week 5, 2/10/2021

Task #	Description	Assigned To
1	MobileNet Version2 research	Sai Arkar Khant
2	MobileNet Version2 research	Anh Minh Ngo
3	GPS To NVIDIA	Sean Atangan
4	Begin building user interface Tkinter	Eric Escobedo
5	GPS To NVIDIA	Eric Escobedo

6	Upload test code to GitHub	Anh Minh Ngo
7	Record dimensions of parts	Sean Atangan
8	Update frame draft	Sean Atangan
9	Upload mobilenet starter code GitHub	Sai Arkar Khant
10	Put Honda civic pictures google drive	Anh Minh Ngo

- Week 6, 2/17/2021

Task #	Description	Assigned To
1	MobileNet Version2 research	Sai Arkar Khant
2	MobileNet Version2 research	Anh Minh Ngo
3	GPS To NVIDIA	Sean Atangan
4	Implement new features to UI	Eric Escobedo
5	GPS To NVIDIA	Eric Escobedo
6	Upload test code to GitHub	Anh Minh Ngo
7	Record dimensions of parts	Sean Atangan
8	Update frame draft	Sean Atangan
9	Upload mobilenet starter code GitHub	Sai Arkar Khant
10	Put Honda civic pictures google drive	Anh Minh Ngo

- Week 7, 2/24/2021

Task #	Description	Assigned To
1	MobileNet Version2 Installation Guide	Sai Arkar Khant
2	MobileNet Version2 Transfer Learning	Anh Minh Ngo
3	Beginning Stages of Neural Network	Sean Atangan
4	Beginning Stages of Neural Network	Eric Escobedo
5	MobileNet Data Integration	Eric Escobedo
6	Upload test code to GitHub	Anh Minh Ngo
7	Record dimensions of parts	Sean Atangan
8	Update frame draft	Sean Atangan
9	Download and run test for MobileNet V2	Sai Arkar Khant

- Week 8, 3/3/2021

Task #	Description	Assigned To
1	Build frame on CAD (solidworks)	Sean Atangan
2	MobileNet Version implement classes and variable	Sai Arkar Khant
3	Download and run test for MobileNet V2	Sai Arkar Khant
4	MobileNet Version2 Transfer Learning	Anh Minh Ngo
5	Upload test code to GitHub	Anh Minh Ngo
6	Begin Custom Class Implementation	Eric Escobedo
7	Integrate Custom Library with MobileNet	Eric Escobedo
8	Run python code test on Nvidia Nano Jetson	Sean Atangan

- Week 9, 3/10/2021

Task #	Description	Assigned To
1	Secondary Class integration	Eric Escobedo
2	Begin Training Module	Eric Escobedo
3	Camera UI python research and building	Sai Arkar Khant
4	Connect to main python code for Camera UI python code	Sai Arkar Khant
5	Build frame on CAD (solidworks)	Sean Atangan
6	Yolonet Integration with mobilenet	Sean Atangan
7	100 images of White 2008 Grey Toyota Corolla for training dataset	Anh Minh Ngo
8	Download and run test for MobileNet V2	Anh Minh Ngo
9	Run python code test on Nvidia Nano Jetson	Sean Atangan

- Week 10, 3/17/2021

Task #	Description	Assigned To
1	Camera UI building (Continuation)	Sai Arkar Khant
2	Camera UI connecting with camera and test	Sai Arkar Khant
3	Custom Image importation with MobileNet	Eric Escobedo

4	Custom Class Integration for Cars	Eric Escobedo
5	Begin Training module	Anh Minh Ngo
6	Add additional class to mobileNet	Anh Minh Ngo
7	Finish Solidworks Draft	Sean Atangan
8	Send .slt file to makerspace	Sean Atangan
9	Implement GPS functionality	Sean Atangan

- Week 11, 3/31/2021

Task #	Description	Assigned To
1	Accuracy Extraction for MobileNet	Eric Escobedo
2	Additional class implementation	Eric Escobedo
3	Final draft for Solid works	Sean Atangan
4	Operation without mouse and keyboard	Sean Atangan
5	Continuation of Camera UI testing	Sai Arkar Khant
6	Object Detection with Yolonet	Sai Arkar Khant
7	Object Detection with Yolonet	Anh Minh Ngo
8	Save the objects as a picture to a folder with Yolonet	Anh Minh Ngo
9	Main function	Sean Atangan