Senior Design Project Proposal
Mar.io
Problem Analysis and Requirements

Jared Balakrishnan, Sarthak Babbar, Akshar Amin
Sai Talla, Arjun Sachdeva, Abhiroop Verma

Advisor: Dr.Matthew Stamm

Department of Electrical and Computer Engineering
Drexel University

November 8, 2019

Contents

List of Figures

List of Tables

Executive Summary

With the meteoric rise of research activity in the area of Computer Vision, numerous applications have surfaced in the world, ultimately leading to a burgeoning multi-billion dollar market. The number of problems being solved in this area, ranging from simple handwriting recognition to autonomous driving has brought to our attention the extent to which human life could be simplified with the help of such technology, thereby helping us eliminate costs as well as leave a smaller carbon footprint.

Mar.io aims to solve a segment of the problems in the path detection area for autonomous system by building a small-scale version of a computer vision system which when iterated in large scale can be used in the areas of robots used for autonomous delivery as well as wheelchair assistance for the disabled.

The project aims to build technology that is committed to improving the quality of life, whilst respecting the communities in which it would be used by striving to make computer vision more simpler and accessible.

Mar.io is designed to detect traversable areas on sidewalks in addition to attempting pathway recognition by building a machine learning model that is trained upon sidewalk images fed through a microcontroller. The machine learning model is intricately composed of components that specialize in collecting, cleaning and analyzing the data prior to using it for the purposes of training and inference. Once this task of inference is successfully achieved, the microcontroller system would be able to perform this detection in real-time. This can then be viewed live on a web application. The entire model is slated to be built using the Python programming language and the libraries associated with it, whereas the web application is slated to be built using a mix of JavaScript and React.

The prototype of *Mar.io* is also designed to be extremely cost-efficient, with the bare bones prototype costing less than $600 and the large-scale business prototype projected to cost a little less than 2 million dollars over a period of two years.

1 Introduction

1.1 Background

The advent of cutting edge technology has changed the very pace of human life; earlier, tasks as simple as running errands required human beings to move from one point to another. But today, the very way in which we see these mundane tasks have changed, for automation and applications on handheld devices have now brought these services to our very fingertips.

Considering the case of any customer-vendor system for delivery of food or mail, it is observed that after the customer utilizes the system to make an order, there is an entire network of human beings handling various tasks before the customer's order is fulfilled by the vendor. In the case of mail and package delivery, as seen in something like Amazon, there is a huge network of delivery systems that employ thousands of human beings as well as vehicles to fulfill delivery deadlines. Same goes for something related to the delivery of food from restaurants, as could be seen in the case of examples such as Caviar or Uber Eats.

On a macroscopic scale, this translates into large amounts of manpower used to perform mundane tasks, large amounts of money spent on things other than the actual product being delivered as well as a lasting carbon footprint in the long run.

Consequently, there is a need of developing autonomous systems powered by renewable sources of energy that help in increased automation of such tasks. Examples of such autonomous systems in the present world include robots, drones and cars although the degree of autonomy among these systems varies . This resulting minimized human intervention translates into a lot of benefits, ranging from economic to environmental.

A lot of companies have already iterated this problem on various scale, producing tremendous results, ranging from Tesla for autonomous cars to Starship for delivery robots. Something that is common across each one of these autonomous systems is a computer vision system that powers the system in helping it performing the task at hand.

More importantly, it was realized by the team behind *Mar.io* that a small scale computer vision system which could aid in solving problems associated with the classification of paths on sidewalks taken by autonomous systems could be applied in a variety of applications ranging from delivery robots to automated wheelchair assistance for the disabled.

1.2 Stakeholders and Needs

Mar.io has a diverse set of stakeholders and the variety of needs that these stakeholders bring along with them to the table. The various stakeholders that are possibly affected by the project include:

- **Drexel University**: The *Mar.io* team strives to commit to Drexel Engineering's mission by committing themselves to the discovery as well as applications of technology that can promote economic development and improve the quality of life.

- **The *Mar.io* Team**: The *Mar.io* team is committed towards conducting research and developing software that can simplify the field of computer vision, thereby making it accessible to anyone and everyone.
- **The University City Community**: *Mar.io* aims to helping the community in positive ways, by ensuring that the implementation of this project shall respect the needs of the community and that no ethical principles will be left unturned.

1.3 Mission Statement

Taking into account the potentially numerous ways in which computer vision systems could be used to actively help in making the world a better place, the mission statement for the project is as follows:

"Mar.io strives to utilize the synergy between Man and Machine to empower each other, making our world a better place for our future generations."

2 Methods

For this project, we plan on collecting video data from a camera mounted on a remote controlled car, and then saving the data as individual frames on a local server. From there, we will clean up the data using Python and some external libraries like OpenCV. Frames will then be labeled using a software program written in Python and its dependent data analysis libraries. The labeled images will be stored on an AWS server. These labeled images will then be fed forward to a pre-trained convolutional neural network (CNN) implemented in Keras. We will fine tune this neural network to distinguish between sidewalks and non-sidewalks areas on a sidewalk. Once the CNN model is ready then this will be tested on a Raspberry Pi to classify if the path is sidewalks or not sidewalks.

3 Design Description

3.1 Specifications and Standards

Below shown is a high level Gane-Sarson data flow diagram for the proposed computer vision system that forms the functional backbone of *Mar.io* :

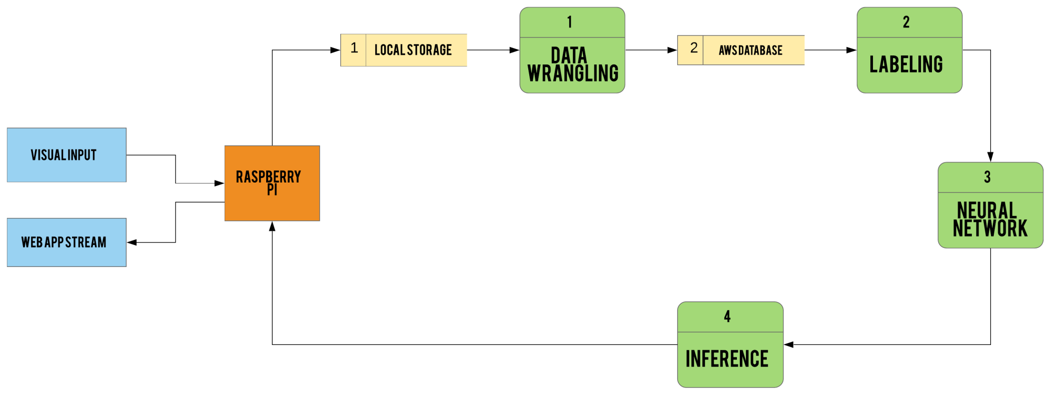


Figure 1: High Level Gane-Sarson Data Flow Diagram of Mar.io

The Raspberry Pi is built onto a small buggy with a camera on top through visual input is streamed into the system. Once the logging of data begins, this data is then stored locally, in a MicroSD card. Once the desired amount of data has been collected, the data present in local storage is picked up by a custom-made data wrangling system which is completely implemented in the Python programming language.

The very purpose of the Data Wrangling System (DWS) is to help with the manipulation of data which enables the discovery of relationships as well as regularities among the elements comprising the dataset. Once the video footage is passed over to the data wrangling system from the data collection system, it is cleaned in order to enable further analysis as well as convert available data into some uniform and consistent format.

Before the functioning of the cleaning process can be explained, it would be a great idea to recall what frame rates mean. Frame rates, often called frames per second or just FPS, is a measure of the frequency at which consecutive images called frames appear on a display. So, if a camera specification reads 50 fps, it means that 50 consecutive images are captured within one second.

The camera used in the microcontroller employed by Mar.io for the purposes of data collection functions at 24 frames per second. This means that every second of video footage recorded would be comprised of 24 consecutive images. Considering the fact that approximately 12 hours of video footage is to be recorded in order to generate a proper trading dataset for the machine learning model, the number of frames generated in the process of data wrangling would amount to approximately 103,7000.

So the video footage collected is handed over to the DWS which condenses every single video into individual frames, which happen to be images. Once all of the footage is converted into images, they are resized into a uniform dimension for the sake of maintaining consistency while analyzing the data. In addition to resizing, a certain amount of feature engineering will be carried out to optimize the output of the DWS.

The processed output from the DWS is then subject to a gargantuan labeling process, with the aim of creating labeled training data that further parts of the system will seamlessly be able to work with.

The computer vision system being developed is a supervised learning problem. For any supervised learning problem, for every corresponding input exists a corresponding output. Similarly, in this case, images comprise the inputs and the output is whether the path is traversable or not. So to create the outputs the team has to label the images.

After all of the frame data is effectively labeled, it is stored in an Amazon Web Services database on the cloud from where it can be easily accessed by the other parts of the system that need to work with the data, as well as the developers working with the data.

It is also to be noted that a significant amount of labeled data (103,7000 images accounting to 12 hours of video footage) will serve as the training data upon which the machine learning model, which will be discussed subsequently, will be applied. This is done so that whenever new, unlabeled data is presented to the model, a likely label can be guessed or predicted for that particular segment of unlabeled data.

Then comes the task of training a convolutional neural network (CNN) with these labeled images so that decisions could be later made with the inference system.

Convolutional Neural Network is a type of neural network which is mostly used when a problem is related to computer vision. CNN best work with image dataset A CNN usually has 2 parts. First part is the feature learning. In feature learning the important features are extracted from a complex multidimensional image vector. This reduces the complexity since the matrix dimensions reduce. The second part of the CNN is the classification system. After features of the images have been extracted, the classification layer, which contains perceptrons, classifies the image.

For a CNN to perform well it has to be trained on a very large dataset. So for this problem a pre-trained CNN model will be used. A pre-trained model can be tuned to fit a similar dataset. This is called transfer learning. There are different aspects to transfer learning. The one will be used here will be re-training few of the layers so it better fits the new training dataset.Few pre-trained models models available include VGG19,Resnet, and GoogleNet.

After the training of the CNN model is accomplished, it is then tested. The testing will be done on an embedded system. Images frames provided to CNN which will output the classification of the image. The classification output would be whether the path in front is walkable or not walkable. This inference has to happen in very short time so that decision making is in real time.

The classification output and the camera view will be streamed to the user from the embedded system. The streaming will be done using a webapp. The webapp will be created using the technologies such as JavaScript and React on the frontend and Python tools such as Flask or Django on the backend.

3.2 Detailed Design

3.2.1 Operating and Development Environment Requirements

Front End-Server-Database

The team plans to use a collection of technologies to make this project seamless and efficient. For the vision system to be correctly executed, the team must develop a way to stream the video feed from the robot itself to a visual device through a server. Considering this, the team will use a front-end tool like React JS to develop a user interface where a live camera feed is displayed. React is a fast and simple tool to make a beautiful UI. If the team plans on creating a mobile app to display a live feed, it will be easy to translate React JS to React Native. React Native uses Javascript code and compiles it to native iOS and Android code making it a seamless transition. Secondly, it will be necessary to set up a server-side of this application. For this, the team will use Django to create the web framework. Django can be used seamlessly with React JS and will provide powerful results. Finally, during the testing and classifying stages, the team will use AWS to store image matrix. This will be necessary to efficiently store data while the team classifies all the frames that will be collected.

Data Collection

To develop an efficient and robust vision system a considerable amount of data would have to be collected. For this project the data would be in the form of frames collected from a camera connected to a raspberry pi, the team chose this configuration of hardware because the same configuration would be used to test the vision system. To collect a diverse dataset and avoid overfitting, the team has selected three different sidewalks, furthermore the data would be collected under different lighting conditions over a period of a few weeks. Hence, four hours of data would be collected from each of the three sidewalk for a total of twelve hours of data. This would result in 103,7000 image frames being recorded, the data would be stored on a local server for further processing.

The team will then use python to analyze the dataset to recognize and amend any outliers and frame errors. Then the individual frames would be resized and reformatted before they can be labelled. As shown in the figure below each image frame will be split into three using opencv before they can be labelled. To accurately label all image frames the team will develop a custom tool using python and Django, then use the tool to label images in specific batch sizes.

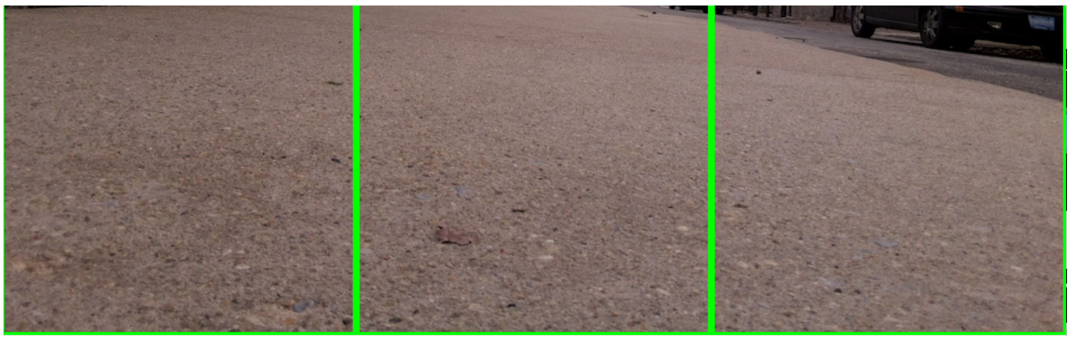


Figure 2: Proposed Dissection of Frame Data for Labeling

The dataset and the labels would then be split into training and testing sets before they can be used for training a convolutional neural net. In order to develop a robust classification model the team has decided to tune an existing classification CNN that has already been trained on various datasets. To achieve this the team would retrain an existing dataset using keras with tensorflow backend.

3.2.2 Convolutional Neural Network (CNN)

A neuron constitutes the basic unit of a neural network.

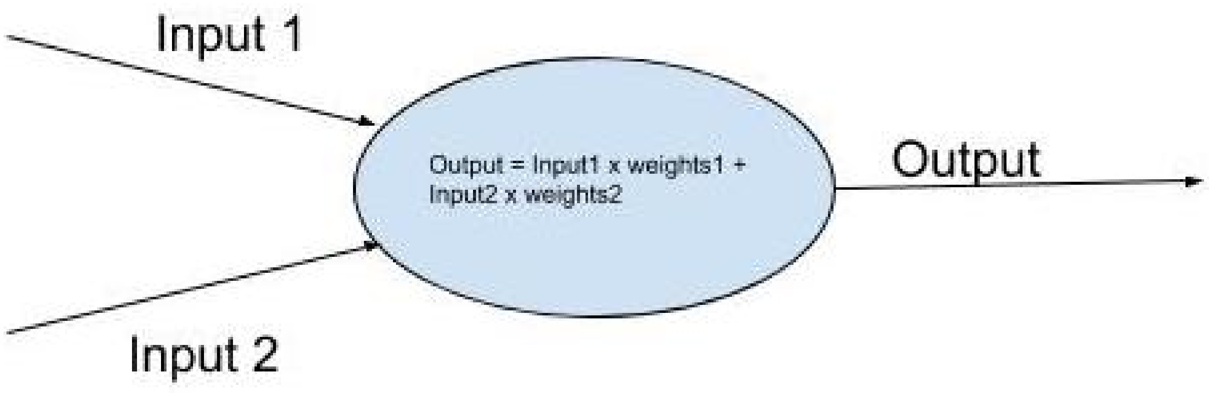


Figure 3: Basic Unit of a Neural Network

For some input a neuron calculates the output. The output is calculated by doing a dot product of the input value and weight associated with that value. A neural network comprises of many layers where each layer has multiple neurons. A basic neural network can have the following layers :

- Input Layer
- Hidden Layer
- Output Layer

Input layer has the same number of neurons as the inputs from the dataset, output layer can have any number of neurons. The output layer depends on what kind of output is required. There can be multiple hidden layers and for each layer there can be different number of neurons. More complex task might require more neurons.

If a neural network is trained on an image then the model might not perform well. This is because a neural network learns the relation between an input and an output. Since images have patterns in them a different kind of neural network is used which is called Convolutional Neural Network (CNN).

A CNN takes an image and applies filters on it. The filters are initialized randomly and during training it gets updated. The process of applying different filters is known as convolution. Once all the filters are applied and the data is normalized the output of the convolutions is sent to neural network. The neural network then outputs final classification for the image.

In this project the CNN is being used to classify data. Given an image to the model it will classified whether the path is walkable or not walkable. Making a good CNN requires lot of data and computing resources, so to solve this problem a pre-trained CNN will be used. A pre-trained CNN can be re-trained with less resources and images to fit the slightly different dataset. The few pre-trained models available are :

- MobileNet
- VGG19
- InceptionV3

3.2.3 Classification Model Testing

To test the model, the team will collect additional image frames from different sidewalks. This dataset will be provided to the model as an input for binary classification. To enable the vision system to be mobile and be used outdoors, the model and other necessary software utilities would be implemented on a low-power embedded system.

3.3 Challenges and Solutions

The challenges that were initially faced were finding methods to physically gather data. To resolve this issue, an RC car and raspberry pi were used as the main platform for the camera. Future challenges include collected data in different situations to ensure that the final result of the vision system contains as much unbiased data as possible. The different situations include varying time of day, location, and weather. Training the vision system is ideal in clear conditions during the day. However, when there is rain or snow, then the data will get skewed and inaccurate resulting in improper training of the vision system. Training the vision when it's dark outside can also skew the results of the data because there isn't enough light for the camera to capture reliable images. The vision system will be trained at different locations, like parks, to increase adaptability for the vision system. Another major problem that is a concern in the future will be tuning the convolutional neural network for classifying the images collected by the vision system. Also, when hundreds of thousands of images have been collected, a software to label these images has to be developed to distinguish the different parts of the image.

4 Context and Impact

4.1 Economic Analysis

The computer vision market has turned out to be one of the fastest growing markets in the world as of now, given the rate at which research is churning out applications to be used in a variety of fields.

In relation to our research, it is best to bifurcate the computer vision market according to the following diagram:

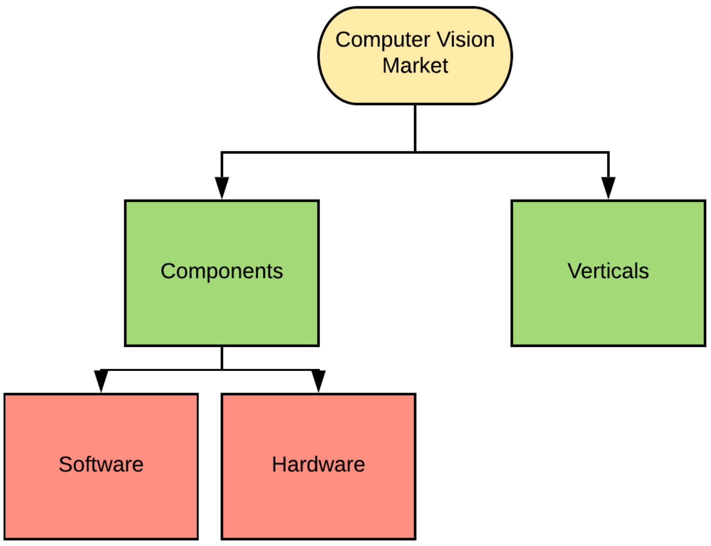


Figure 4: The Computer Vision Market

Componentwise, the computer vision market is best bifurcated into the hardware and software markets.

The hardware market has emerged to be the dominant player in the computer vision market, primarily owing to the advances in computer vision research that have allowed for lower costs of devices such as cameras, personal computers and mobile phones. With computer vision research churning out more and more applications day by day, hardware companies as well as the commercial sector as a whole have made it a priority to incorporate these technologies in hardware at a pace never seen before.

The software market, on the other hand, has set its own precedent by exhibiting the fastest growth rate. The measure of performance utilized here is called **Compound Annual Growth Rate**, or *CAGR* for short.

Before anything can be mentioned regarding the various CAGR figures of the computer vision market, it is imperative to understand what this term means. CAGR is defined as being that rate of return which is to be observed in an investment, so that it can grow from its beginning balance to its ending balance provided that the profits are always reinvested at the end of each year during the investment's life cycle.

Market research has shown that the computer vision market as a whole is slated to grow at a CAGR of **31.65%** in the forecast period ranging from 2017 to 2023. During this time span, the market cap of the computer vision market is supposed to approach a staggering **USD 48.32 billion**. The hardware segment of the market was valued at **USD 4.7 billion** in 2016, and is expected to reach **USD 26.43 billion** by 2023, with a CAGR of **29.19%** whereas the software segment is expected to reach **USD 21.89 billion** by 2023 at a CAGR of **35.13%**.

The high value observed for the CAGR of the software component in the computer vision market is primarily attributed to the pace at which research is enabling numerous applications ranging from artificial intelligence to augmented reality and neural networks.

The multitude of applications made possible by computer vision research makes it imperative to define what horizontal and vertical markets are, prior to discussing the specifics about the economics of Mar.io in relation with this market research.

A vertical market is defined as one that is comprised of very niche players. These niche players usually strive to serve very specific needs: that is, they cater to a customer set that identifies itself in a very narrow industry. This in turn implies that the goods and services sold in vertical markets tend to be very similar, thereby encouraging competition among the market players.

On the other hand, a horizontal market usually comprises of goods and services which enable a platitude of businesses: that is, in the simplest terms, they cut across a series of vertical markets.

Given the fact that Mar.io aims to solve a problem within a specific section of the computer vision industry, it makes more sense to devote more attention to the vertical markets rather than the horizontal markets.

To name a few, the various vertical markets that are contained within the computer vision market include:

- Agriculture
- Automotive
- Healthcare
- Entertainment
- Consumer Electronics

The automotive segment has clearly emerged to be the leading player in the vertical markets, be it in terms of market value or technological innovation. This is manifested in the very way in which the automotive sector has been revolutionized by the emergence of applications supplemented by computer vision: previously, the automotive sector stagnated in terms of growth with no diversity among the landscape. But now, with technologies such as autonomous driving (as could be seen in the case of the precedent set by Tesla), traffic sign detection, lane departure warning to name a few, there is no more room for stagnated growth.

The automotive segment has led the vertical markets by exhibiting a CAGR of **33.86%** in the forecast period of 2017-2023 with a market value maintained at **USD 1626.7 million**.

Considering the fact that Mar.io does not have a lot of moving components with the exception of the hardware used in the process of data collection, the costs associated with prototyping can be considered negligible, with the bullion share of the costs being associated with the microcontroller and the cloud services used to hold all of the data.

Keeping in view the burgeoning pace at which the computer vision market is growing, the long-term economic goals associated with Mar.io differ from a typical product in that the aim is to simplify the technology that allows minimal hardware to be used, thereby cutting costs at a significant level.

4.2 Environmental Impact Analysis

The computer vision system is a software program which has applications in various fields. One area is food and package delivery. Current delivery services have huge carbon footprints which can have a negative impact on the environment. A delivery robot can help reduce these carbon footprints. There are some external impacts of building a vision system. As time progresses, more data will be collected, which is going to be stored in the cloud. Also, the CNN will get progressively more complex, and as a result, more computing resources will be needed to train it. These resources will utilize cloud as well. Cloud computing does not directly impact the project but the data centers can have an impact on the environment.

4.3 Social Impact Analysis

The vision system is going to substantially help optimize delivering items within a certain distance through autonomous delivery robots. Big areas of impact include college campuses where demand for delivery is extremely high. Students can easily use these autonomous robots to have food, packages, books, or other items delivered directly to any location specified for an extremely cheap price. A great vision system will increase speed and efficiency in delivery for these robots. This will increase convenience for students to have items delivered to them using these innovative machines rather than manually going to local stores to buy items themselves.

There may be concern from many people that the robots cannot handle every situation that may be presented to them. This is where the command-and-control center plays a key role. A command-and-control center will be monitoring these robots and will step in if a robot is facing any unforeseen circumstances. For example, if a robot is in route for delivered and is stuck at a construction site or is accidentally hit by another object that would impair it from completing the delivery, then the command-and-control center will step in and help the robot. Obviously, introducing autonomous robots to human kind won't be an easy thing for humans to cope with because the thought of that might seem too unnatural for some people. Starship Technologies faced a similar problem when they were making their Starship robots. The company designed their robots to travel at a maximum speed of only 4 mph to avoid frightening people on the streets. The vision system will be designed to support these kinds of robots.

There may be concern from people that these autonomous robots might hinder their daily lives, but the robots are supposed to fit right in with human life. By design, the robots are made to blend in with the infrastructure of cities, neighborhoods, campuses, and suburbs to almost go unnoticed. The vision system will promote greater intelligence for these robots to help withhold the principles of this design and ensure that they thrive alongside humankind.

4.4 Ethical Analysis

There are a lot of ethical impact considerations to make while working on this project. With an autonomous vision system and a robot, there are numerous ethical dilemmas that form. First and foremost, the autonomous robot will have a camera which means it is taking in a constant video of people and things that cross its path. It is unfeasible to get permission from everyone or everything that crosses the robots path while the robot operates. There are laws that allow for recording of public areas without focusing on the people itself.

However, there are multiple laws that can be violated if the team isn't careful in the implementation. Since the robot is small and is likely to be low to the ground. The camera gets footage from angles that aren't normally expected. Some behaviors like upskirting come into question if the camera is capturing things at an improper angle. Upskirting and downblousing are criminal offenses, regardless if it is unintentional. The team must make sure that the camera is pointed directly perpendicular to the ground and the camera is grabbing footage of just the ground.

There are not many other ethical impacts of autonomous food delivery robots. Unlike autonomous cars, these robots are small and have very little impact on people's health and accidents that may happen. Debates like the "trolley car" problem aren't relevant to this robot due to its size and impact.

5 Project Management

5.1 Team Organization

The intiative behind *Mar.io* is spearheaded by six students from the Department of Electrical and Computer Engineering, and advised by **Dr.Matthew Stamm** from the same department. **Dr.Edward Kim** from the College of Computing and Informatics serves as an external consultant, providing the team with advice on approaching the software design involved in the project.

The members of the Mar.io team, listed in alphabetical order, are:

- Abhiroop Verma
- Akshar Amin
- Arjun Sachdeva
- Jared Balakrishnan
- Sai Talla
- Sarthak Babbar

From an organizational standpoint, the team is very flat. The rationale behind instituting a flat hierarchy stems from the fact that every member is fired about the vision being pursued, and a very good way of making it happen together is by giving them a considerable amount of autonomy while working towards the same end goal.

Also, considering the fact that the members of the team are looking for different kinds of challenges in the professional arena after graduation, they are assigned one area of their choice to specialize in, on top of being involved firsthand in the development of the software that powers the project. This in turn culminates in the creation of the following roles:

- *Software Engineer, Machine Learning*: The software engineer in charge of Machine Learning is tasked with the challenge of coming up with creative as well as effective solutions to implement the decision making system that powers the computer vision system that Mar.io is.
- *Software Engineering, Embedded Systems*: The embedded systems engineer is entrusted with the extremely important task of designing the hardware system required for the purposes of data collection, in addition to making sure that the software and hardware are interfaced in the most fluidic manner possible. This role is tantamount to the success of the project, given the fact that the hardware needs to handle video input as well as the inference output from the machine learning system. This engineer will also be tasked with building further iterations of the system if time permits.
- *Software Engineer in Test*: The software engineer in test is tasked with validating all of the functional and nonfunctional requirements of the system, in addition to ensuring that the entire system works as one cohesive unit. Tasks include developing unit tests for all the modules and processes that comprise Mar.io.

- *Data Scientist*: The Data Scientist works in tandem with the Data Engineer to ensure the integrity of the data used in the system. Given the fact that the system needs to handle a large amount of data ($\tilde{1}$ million images), the Data Scientist ensures that all of the data being analyzed is of a consistent format and doesn't suffer from any issues like missing data and the like.
- *Data Engineer*: The data engineer supplements the work of the data scientist by ensuring that the system holds a viable data pipeline to handle the amount of data passing in and out of it. This role is also tasked with making executive decisions on the flow of data through the entire architecture of the system, while also working in unison with the data scientist.
- *Product Manager*: The explicit goal of the Product Manager role is to partner with the software development team and working through the entire product cycle ranging from software development to ironing out specifications, as the advocate for the end-users. A key benefit to adding the Product Manager position is how far more agile the team tends to be, working as a cohesive unit.

The assignment of the above mentioned roles to the members of the team is as shown in the table below:

Table 1: Designated Roles for Mar.io Team Members

Name	**Role**
Abhiroop Verma	Data Scientist
Akshar Amin	Data Engineer
Arjun Sachdeva	Software Engineer, Embedded Systems
Jared Balakrishnan	Product Manager
Sai Talla	Software Engineer in Test
Sarthak Babbar	Software Engineering, Machine Learning

5.2 Organizational Toolkit

Considering the diverse experiences gained by the members while working in different firms in the industry, it was decided that this project would be run like one by a company in the technology industry. This translates into adopting software engineering industry methodologies such as Agile development and workflow tools such as Notion and Slack, to name a few. Other practices incorporated include two all-hands meetings per week and a dedicated software repository for the purposes of version control using the Git technology.

The technology stack used in the organizational level for different purposes can be seen in the table shown below:

Table 2: Organizational Tools used by the Mar.io Team

Team Member	**Compensation in USD**
Intra-Team Communications	Slack
Graphic Design	Sketch
Workflow Scheduling	Notion
Software Documentation	GitHub
Typesetting	LaTeX
Knowledge Base Repository	Notion, Inkdrop

5.3 Schedule and Milestones

Keeping in mind the Agile software development manifesto and the ideal product development life cycle, the development of the project is broken down into four major pillars, as could be seen from the infographic shown below:

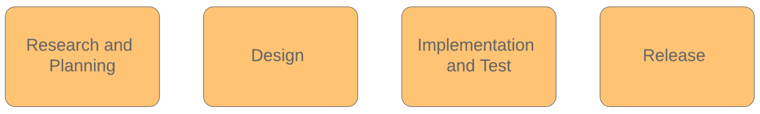


Figure 5: Componentwise Organization of the Project

The **Research and Planning** phase is the first step of the product's life cycle. This phase could broadly be defined as the time when the team comes together to decide what is to be built. The various factors influencing what is to be built stem from factors such as competitive analysis, new technology, brainstorming or the big vision for the software product.

In terms of *Mar.io*, this phase plays a big role, given this is where the roadmap is proposed and subsequently created. This involves figuring out a long term plan for the team. All possible sources are consulted in order to create a large list of potential features or development tasks. Depending on factors such as the competitive landscape, availability of time and the team's expertise, the features and scenarios are prioritized.

Once the development tasks and feature set is figured out, comes the task of defining what success necessarily looks like. In the case of *Mar.io*, the team uses an internal model of Objectives and Key Results to communicate the most important goals of the team.

After having figured out what the team is going to build comes the **Design** phase, which could possibly be considered to be the most important phase in the product life cycle. It is during this phase that the features and the functionality of the product is defined.

In the case of *Mar.io*, the team follows a model of ironing out a detailed functional specification, the fine tuned version of which is basically what constitutes this project proposal. The team sits down together to talk about the goals of the project, and brainstorm on a whiteboard, and then iterate by giving feedback on the proposed design. The resulting functional specification is then inspected, reviewed and iterated on by the advisors as well as the team, with the team being entrusted with the responsibility of making every product-related decision. In the words of Dr.Stamm, this is defined as a time of "*repeatedly poking holes until it seems right*".

Owing to the various time constraints, it was decided that these two phases of the product life cycle would run concurrently.

During the **Implementation and Test** phase, the developers begin to write the software that powers the project.

If in case it turns out that the implementation of a certain part of the system is harder than anticipated, the team comes together to change that part of the system in a way as to make it easier to implement.

As implementation of the code base proceeds, a phase of testing concurrently begins. The purpose of testing is to make sure that every single part of the system works in the way in which it is supposed to, in a cohesive manner. During the testing phase, the team starts to gather feedback and report bugs encountered in the early versions of the product. This is because sometimes, certain features or parts of the system may not work as expected once it is used in a real world scenario.

To find an diagnose problems such as this, the *Mar.io* team will run a series of experiments and do several rounds of internal dogfooding. Originating from the phrase "*Eat your own dogfood*", dogfooding simply refers to the team using the product themselves multiple times and optimizing the product by studying those results.

As feedback and dogfooding results begin to trickle in, the team identifies the most important issues and attempt to iterate on the design to reach a better solution.

The final stage in *Mar.io*'s life cycle is the **Release** stage. When the development process is finally wrapped up, everything is documented very carefully and subject to approval from the advisors as well as the necessary faculty in the department.

Once everything looks ready to go, the formal development process is stalled temporarily, and the product is presented to the appropriate body.

5.3.1 Scheduling

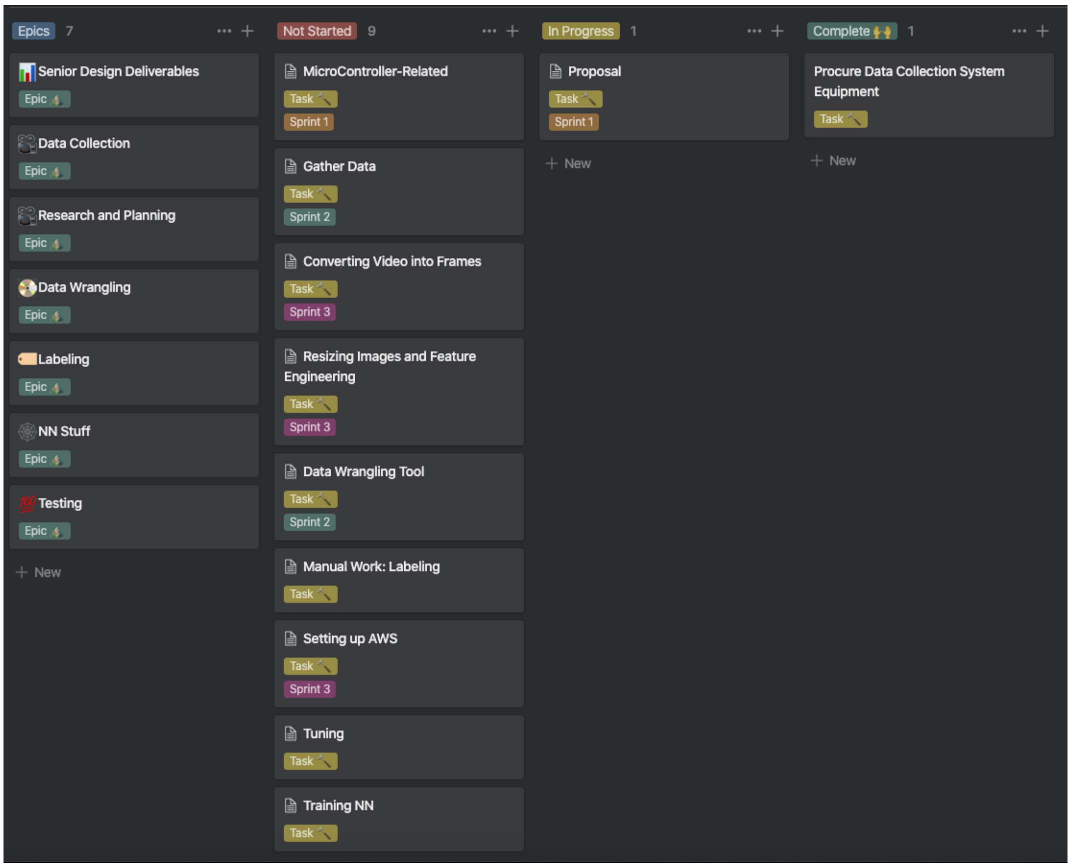
Keeping in mind the way in which software development projects are carried out in the industry, the *Mar.io* team distilled the tasks comprising the different stages of the product life cycle explained above according to the following structure:

- **Epics**: Epics are large overarching initiatives.
- **Sprints**: Sprints are fortnightly time-bounded pushes to complete a set of tasks.
- **Tasks**: Tasks are the actions that make up epics.

- **Bugs**: Bugs are tasks related to fixing things.

The development of *Mar.io* is slated to take place from September 2019 through May 2020. The decomposition of the project into the respective epics, sprints and tasks can be inferred from the below shown infographic. Bugs have not been included in this illustration owing to the fact that the software development process has not started in a fully-fledged fashion at the moment when this proposal is being drafted. Some tasks don't have sprints assigned to them either. This is because of the fact that sprints have only been planned for tasks that are being carried out in the Fall term. The presence of winter break in the midst of the project mandated that the sprints for subsequent tasks be planned later on.

It is imperative to understand that this is the internal timeline that the team wishes to follow. The internal timeline differs from the external timeline in that an emphasis is placed on completing tasks well ahead of time as outlined in the official, external timeline. The internal timeline aims to complete the work related to the project a couple months in advance prior to the actual final presentations as scheduled by the Department of Electrical and Computer Engineering. For the external timeline, please refer to the **Gantt Chart** in **Appendix**.



Epics 7

- Senior Design Deliverables — Epic
- Data Collection — Epic
- Research and Planning — Epic
- Data Wrangling — Epic
- Labeling — Epic
- NN Stuff — Epic
- Testing — Epic
- + New

Not Started 9

- MicroController-Related — Task — Sprint 1
- Gather Data — Task — Sprint 2
- Converting Video into Frames — Task — Sprint 3
- Resizing Images and Feature Engineering — Task — Sprint 3
- Data Wrangling Tool — Task — Sprint 2
- Manual Work: Labeling — Task
- Setting up AWS — Task — Sprint 3
- Tuning — Task
- Training NN — Task

In Progress 1

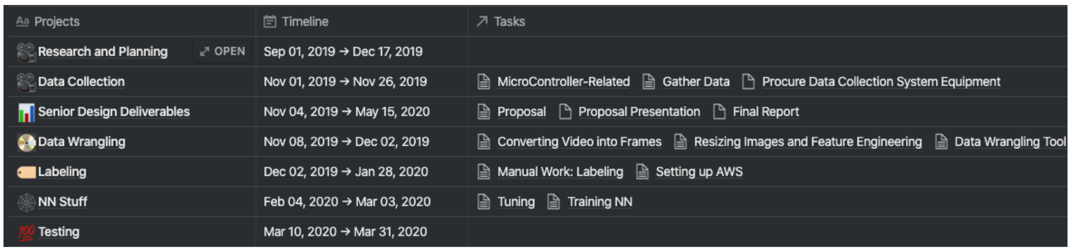
- Proposal — Task — Sprint 1
- + New

Complete 🙌 1

- Procure Data Collection System Equipment — Task
- + New

Figure 6: Scheduling of the *Mar.io* Project

For a more detailed timeline of the various epics as well as the tasks that comprise them, please refer to the following infographic:



Projects	Timeline	Tasks
Research and Planning	Sep 01, 2019 → Dec 17, 2019	
Data Collection	Nov 01, 2019 → Nov 26, 2019	MicroController-Related, Gather Data, Procure Data Collection System Equipment
Senior Design Deliverables	Nov 04, 2019 → May 15, 2020	Proposal, Proposal Presentation, Final Report
Data Wrangling	Nov 08, 2019 → Dec 02, 2019	Converting Video into Frames, Resizing Images and Feature Engineering, Data Wrangling Tool
Labeling	Dec 02, 2019 → Jan 28, 2020	Manual Work: Labeling, Setting up AWS
NN Stuff	Feb 04, 2020 → Mar 03, 2020	Tuning, Training NN
Testing	Mar 10, 2020 → Mar 31, 2020	

Figure 7: Timeline for the Project Epics

5.4 Project Budget

The timeline for this project when carried out on a large scale will extend over the course of two years. The bulk of the costs associated with the project will be incurred on the team working behind the project. Excluding human resources, the other main costs incurred will be from maintaining AWS and SQL storage databases as well as training the model on AWS. The main reason as to why this is so is because of the fact that the project culminates in the development of a software system, with a minimal number of moving parts.

Below shown is the table detailing the breakdown of the costs associated with the workforce behind the development of *Mar.io* on a per-year basis:

Table 3: Manpower Costs Per Year Associated with *Mar.io*

Function	**Tool**
Embedded Systems Engineer	$105,381
Data Scientist	$99,558
Machine Learning Engineer	$94,510
Testing Engineer	$95,981
Data Engineer	$94,572
Product Manager	$125,210
Consultants (x2)	$253,294
Sum Total	$868,686

The breakdown of costs of the software as well as hardware, including data collection equipment, is described in the table below:

Table 4: Equipment Costs Associated with *Mar.io*

Component	**Cost**
AWS Storage	$14,131.20
SQL Databse	$2,382.72
Training on AWS	$2,324.16
GoPro Hero 7 Camera	$400
Camera Stabilizer	$100
Embedded System	$500
Sum Total	$19,838.00

6 References

Convolutional neural network. (2019, October 26). Retrieved from
https://en.wikipedia.org/wiki/Convolutional_neural_network

Adhikari, Ani (2019, September 30). Inferential Thinking. Retrieved from
https://www.inferentialthinking.com/chapters/01/1/intro.html

Jay, P. (2018, April 8). Transfer Learning using Keras. Retrieved from
https://medium.com/@14prakash/transfer-learning-using-keras-d804b2e04ef8

Rules of Machine Learning: — Google Developers. (n.d.). Retrieved from
https://developers.google.com/machine-learning/guides/rules-of-ml

Saha, S. (2018, December 17). A Comprehensiv Guide to Neural Networks. Retrieved from
https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-

Stewart, M. (2019, February 27). Simple Introduction to Convolutional Neural Networks. Retrieved from
https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3

Univeristy of Helsinki: Data analysis with Python - Summer 2019. (n.d.). Retrieved from
https://saskeli.github.io/data-analysis-with-python-summer-2019/

Williams, J. (n.d.). Efficient and Scalable Inference. Retrieved from
https://ps.is.tuebingen.mpg.de/research_projects/efficient-and-scalable-inference

Raspberry Pi Documentation: Camera Module. (n.d.). Retrieved from
https://www.raspberrypi.org/documentation/usage/camera/

Frame rate. (2019, November 5). Retrieved from
https://en.wikipedia.org/wiki/Frame_rate#Film_and_video

Market Research World: Computer Vision Market 2019 - Global Industry Analysis by Trends, Size, Share, Company Overview, Growth and Forecast by 2023. Retrieved from
https://on.mktw.net/32uG3Wf

Machine Vision Market. (n.d.). Retrieved from
https://www.marketsandmarkets.com/Market-Reports/industrial-machine-vision-market-23424673
html

Nahm, S., Busch, C., Sykora, L., & Yen, C. (n.d.). Vertical vs. Horizontal Markets. Retrieved from
https://ecorner.stanford.edu/in-brief/vertical-vs-horizontal-markets/

Murphy, C. B. (2019, October 6). Understanding the Compound Annual Growth Rate – CAGR. Retrieved from
https://www.investopedia.com/terms/c/cagr.asp

Walsh, B. (2014, April 2). New Greenpeace Report Shows the Environmental Impact of the Internet. Retrieved from
https://time.com/46777/your-data-is-dirty-the-carbon-price-of-cloud-computing/

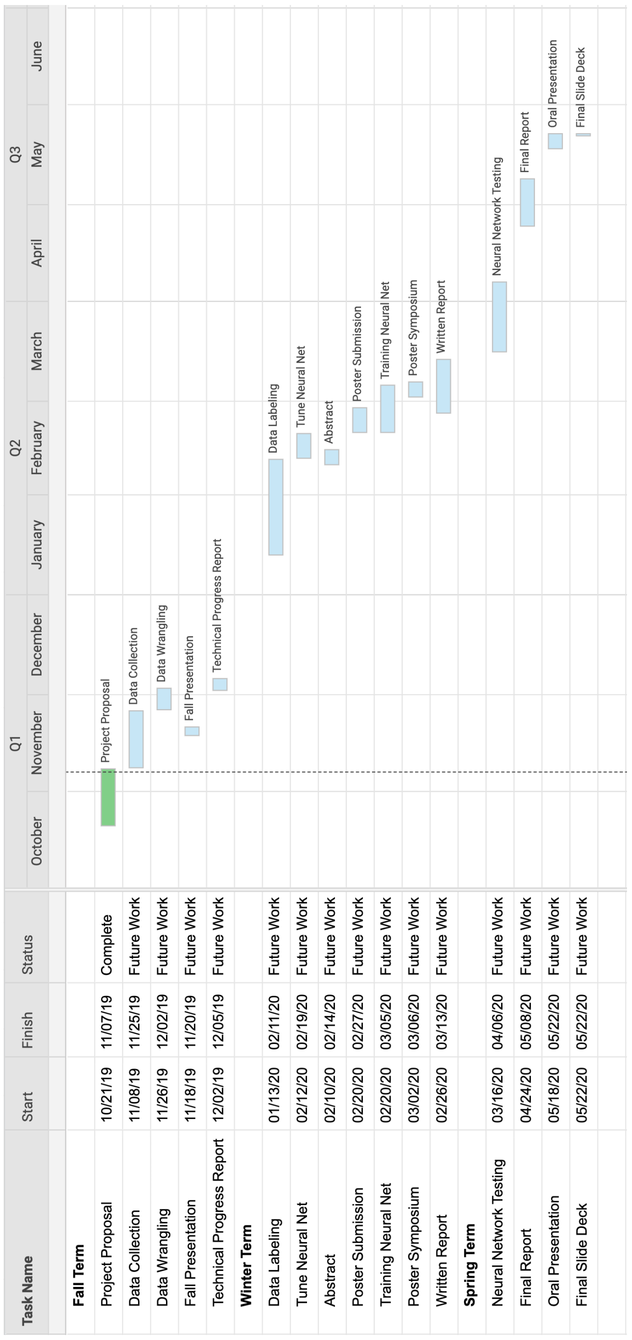
Dormehl, L. (2019, May 22). How Starship Technologies Pioneered The Delivery Robot Model. Retrieved from
`https://www.digitaltrends.com/cool-tech/how-starship-technologies-created-delivery-robots/`

Feldman, A. (2019, September 4). Starship Technologies Raises $40 Million To Expand Its Food-Delivery Robots On College Campuses. Retrieved from
`https://www.forbes.com/sites/amyfeldman/2019/08/20/starship-technologies-raises-40m-to-exp`
`#458d1a3b1cec`

Are you getting paid your Market Salary? (n.d.). Retrieved from
`https://www.smh.com.au/technology/upskirting-to-become-a-crime-20060729-gdo2at.`
`html`

A Gantt Chart

Below shown is the Gantt chart detailing the scheduled, strict time-bound schedule for the *Mar.io* project:



Task Name	Start	Finish	Status
Fall Term			
Project Proposal	10/21/19	11/07/19	Complete
Data Collection	11/08/19	11/25/19	Future Work
Data Wrangling	11/26/19	12/02/19	Future Work
Fall Presentation	11/18/19	11/20/19	Future Work
Technical Progress Report	12/02/19	12/05/19	Future Work
Winter Term			
Data Labeling	01/13/20	02/11/20	Future Work
Tune Neural Net	02/12/20	02/19/20	Future Work
Abstract	02/10/20	02/14/20	Future Work
Poster Submission	02/20/20	02/27/20	Future Work
Training Neural Net	02/20/20	03/05/20	Future Work
Poster Symposium	03/02/20	03/06/20	Future Work
Written Report	02/26/20	03/13/20	Future Work
Spring Term			
Neural Network Testing	03/16/20	04/06/20	Future Work
Final Report	04/24/20	05/08/20	Future Work
Oral Presentation	05/18/20	05/22/20	Future Work
Final Slide Deck	05/22/20	05/22/20	Future Work

Figure 8: Gantt Chart for *Mar.io*