FINAL REPORT FYP

CID (Can It Drive):
Research into Development and Testing of Ethics Engine for Self-Driving AIs, in Pre-existing Virtual Environments

Daniyal Selan 2106268 ||Mahmoud Hamra 2174936

Table of Contents

Table of Figures

1. Introduction

The following section provides an overview of the project with three main questions.

Why this project is important?

In the field of computer science, the biggest revolutions have come about when the power to develop and explore a new technology is given to not only the companies with large resource pools, but also to individual developers, academics and students. This is what made the initial Window's Desktop so powerful and what has allowed it to dominate the PC market till this time, it is because individuals could develop and share their ideas across an easily accessible environment. In recent years, another example of such a revolution is the emergence of Machine Learning algorithms like Neural Networks. Even though Neural Networks have existed in concept since the 1950s, and institutes had implemented examples in the 1980s[1], they have had a recent surge in popularity because they have become relatively easy to access and use. Google and Microsoft both have invested in cloud solutions to allow individual developers to access hardware to implement Neural Networks on, and libraries like Tensor Flow have made it infinitely easier for individuals to experiment with and learn from Neural Networks.

The rise in popularity of Machine Learning has led to a rise in autonomous cars. Projects like Google's Waymo Car project have shown that a completely autonomous car is feasible in the real world. This leads to question on how these cars will be implemented, how they will be tested by institutions and governments, and how these cars will handle the concept of ethical decision making.

We believe exploring a framework or an approach that can make it easier for an individual developer, student or academic to develop and test their own solution is a way to bring about the revolution of ethical self-driving cars. Providing tools and defining an approach which is modular and makes it easier for autonomous and ethical issues to be validated is at the core of our mission.

Autonomous cars are the future, there is a need to solve the issues detailed above and that is why our project is important.

What is the scope of our project?

The project involves researching existing literature review, and exploring and developing solutions that meet the following 3 broad requirements:

1. A framework or approach that can make it easier to develop systems and validate ethical autonomous driving systems
2. A framework or approach that can make it easier to modify and enhance pre-existing ethical autonomous driving solutions already built on the framework
3. A framework or approach that makes it easier to standardize and to test ethical autonomous driving solutions

How we are approaching our project?

The scope of our project is broad, and it is easier if we approach the project by diving into distinct parts. Each of these parts will be discussed in more detail in the rest of the report.

1. Literature Review
2. Designing sub-systems and a framework
3. Simulated environments and their modifications
4. Ethics Engine
5.

Approach for Final Report

For the final report, for the sake of length and readability, we are going to give a high level over view of the work done on the system. We will try not to repeat the details that we already repeated in previous reports. Hence, some of the technical detail which has already been mentioned in the previous reports will not be mentioned here.

2. Formulation of Design Problem

The aim of the project is to explore ways to simplify and lower the barrier of entry for people to develop their own ethical self-driving solutions.

The rationale behind this is the fact that driving is a very perception and context-based activity. There are lots of tiny minutia that needs to be accounted for when driving. Things that cannot be accounted for when developing a driving system solely based on analytical analysis. A system designed to drive must be developed in an environment in which it can interact with freely and in a non-linear manner, in order to see how the system will behave when confronted with every possible interaction that can occur. Waymo's system is developed inside of a virtual environment that was created specifically to test their self-driving system. Everyday Waymo's system drives 8 million virtual miles. Hence, it is important to explore a framework or approach that can make it easier to develop a self-driving system in a virtual environment. Since we do not have access to a virtual environment, we will use a video game which can approximate real life scenarios on the road. This will be detailed in a later section.

Scope of Problem.

The objectives that need to be completed to achieve our aim can be found in the scope of the problem detailed above. The scope can be elaborated upon further by breaking down the problem into 3 distinct parts:

1. A framework or approach that can make it easier to develop ethical autonomous driving systems
    a. Make it easy to collect and pre-process data that is needed to train different AI models in the system
    b. Simplify the inputs and outputs of the sub-systems (this is done so that it is easier for sub-systems to communicate with each other)
    c. Modify existing AI models to work within the system (such as AlexNet and Google's InceptionNet)
    d. Use existing cloud solutions to train the prepared Neural Networks
2. A framework or approach that can make it easier to modify and enhance pre-existing ethical autonomous driving solutions already built on the framework
    a. Have a modular approach when designing the system so sub-systems can be modified and changed easily without having to change the whole system
    b. Standardize the inputs and outputs of the sub-systems so that sub-systems can be changed without having to alter the data set on which that sub-system is trained and validated upon
3. A framework or approach that makes it easier to standardize and to test ethical autonomous driving solutions
    a. Standardizing and simplifying inputs and outputs of the self-driving system and it's sub-systems. (this will make it easier to log the activity of the car at any given point and to see which sub-system acted incorrectly in case of an error or an accident)

We will achieve our objectives and aim by carrying out a case study. If we, students with limited expertise and resources, can develop an approach and use that approach to make and thoroughly test an ethical self-driving car, then our approach can be considered a success.

3. Research Method and Approach

As mentioned above, the goal of this project is not to create the best autonomous car ever, but rather to explore an approach that makes it easier to develop autonomous car solutions. Specifically, we want to facilitate the development of solutions for the problem: How to make an autonomous car act ethically.

To this end, our approach will be to focus on development and research of tools and systems that need to be in place for a car to make ethical decisions. We will not focus on the ethical solution itself.

Our approach to designing the system will be a modular one. We want to make the system easy to understand and modify. A developer should be able to implement his or her own ethical decision making algorithm without having to modify the whole system. We will try to utilize preexisting solutions as component for the system wherever possible. This is so we can enable rapid development of our system, whilst at the same time exploring different architectures and solutions for our system.

To maintain modularity and simplicity, we will try to delegate complexity to individual sub-systems rather than increase global complexity. I.E, each sub-system will be responsible of a highly complex task, the inputs and outputs to the sub-system will be kept simple. This is so the sub-systems can be treated as black boxes and to make it easy for developers to understand and manipulate the inputs and outputs between each sub-system.

We will limit the scope of our autonomous car. I.E, we will only limit our car to be driven in certain conditions (like the city center without the presence of pedestrians and traffic). This is so we can easily collect data and iterate through different versions of the system at a rapid pace

We will continue literature review through out the duration of the project. We want to identify how current autonomous cars are implemented, how they tackle the ethical driving and dilemma, and what constitutes as ethical driving. We will include more relevant literature review as the development of our system continues and expands.

As mentioned above, we plan to implement rapid development of the different sub-systems in our system. We will test each sub-system independently and then try and implement the whole system together and run the system in the simulation.

4. Analysis of Existing Literature

An extensive survey and analysis of existing literature was conducted on the following topics:

- Image processing
- Data normalization
- Feature Extraction
- Autonomous driving solutions
- Ethical approaches to Autonomous vehicles
- Multi-Agent systems
- Inference Engines
- Neural Networks

This section details the literature review and analysis done. It also details how each literature source reviewed is relevant to our project, and what lessons and concepts we are taking from each source

Previous Literature Review

The problem being considered is the how to make development of ethical self-driving cars easier and more convenient for individuals. This means developing a system which can not only navigate roads and avoid hitting objects, but also make the ethical choice when faced with a situation that requires decision making.

Waymo

Waymo is Google's Self-Driving car project, and they are currently licensed to operate completely driverless cars in certain localities in California. As such, they have had to tackle with ethical and practical implications of having AI interact with a densely populated and complex environment. For the purposes of our project, we are going to study the approach Waymo takes to solve the self-driving problem [2].

The director of Engineering Sacha Arnoud, when speaking at a lecture at MIT, gave a broad overview of how the Waymo car works [3].

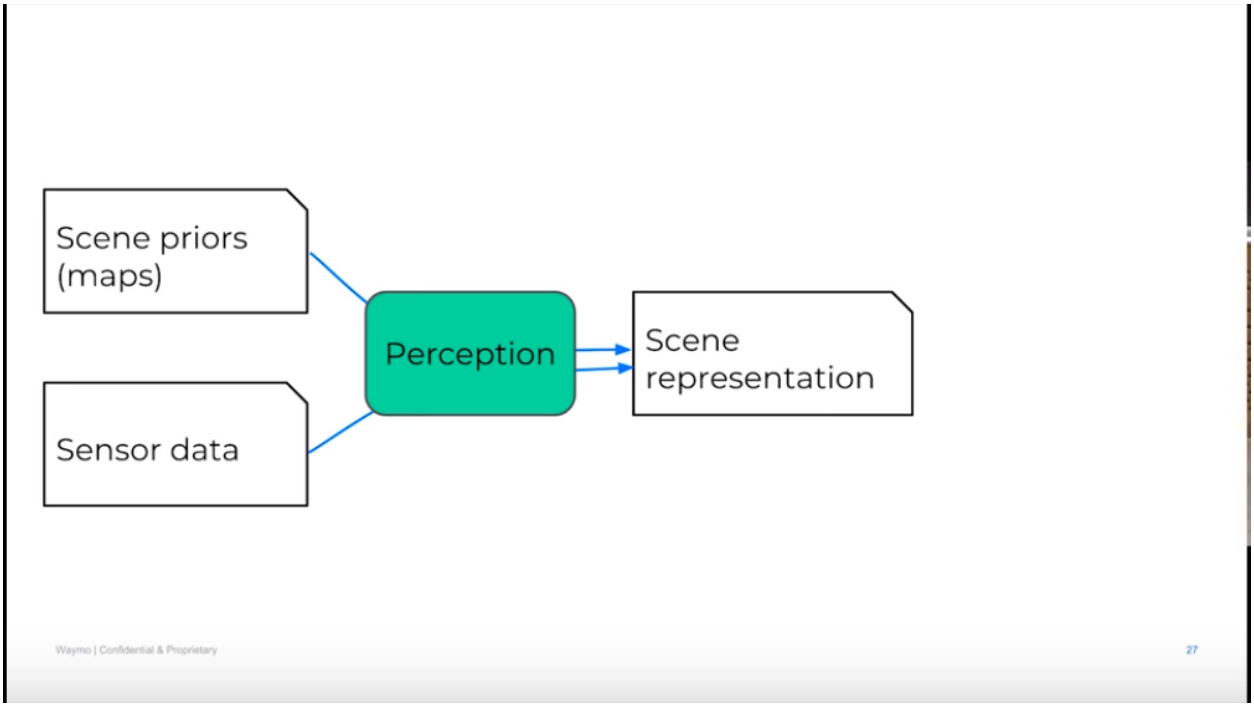

Fig 4.1: Waymo's Perception Approach

The car's system works on 2 types of inputs, already mapped information, such as the layout of streets, and real-time data gathered from the cars sensors, such as radar and lidar. The car uses this information to perceive the world around and create a sense of perception, which it can use to create a representation of the state of the world and then predict the next state of the world, and plot a course to navigate through it safely [4].

Fig 4.2: Waymo Car Navigating the World

The picture above represents all the data collected by the car, layered on top of already mapped data. The car can then plot and adjust a path to follow.

The biggest benefit of this approach is that the car can use different deep learning techniques to identify different entities on the road and predict their likely behavior. This allows the car to be safer than a human. In order to achieve this, the car works with a lot of layered system. These systems need a lot of

data to be trained and have to interact with each other in complex ways. The final output is the path that the car needs to follow. These systems leverage machine learning algorithms to learn how to carry out their tasks.

The downside of this approach is the sheer complexity of it and the huge amount of data that needs to be collected. The data also needs to be preprocessed to ensure that the car is trained to behave correctly. Additionally, for the car to have a sense of perception, it needs to have some sort of temporal memory, i.e, it remembers what the previous inputs were. This adds complexity to the system and makes it harder for the data to be collected and preprocessed. Essentially, the only people who can afford and manage to execute a self-driving car this way are those with deep pockets and plenty of infrastructure. For this project we can still use some of the techniques and algorithms that waymo uses in its car, but porting their solution whole sale to work hardware accessible to us is not feasible. Even if such hardware was available, we do not have the time or expertise to gather the data for their system to work. Google has sent billions on projects like google earth, which allows them free access to such data, and have multi-million dollar server farms to store and process that data. We have access to neither.

Approach to ethical decision making

A goal for the project is to implement a basic ethical decision-making system, and to implement a way to train such a system. Johannes Himmelreich, a researcher at Stanford, identifies the fact that ethical decisions must be made in mundane situations as well as emergency, extra-ordinary circumstances. An example that he provided is the way in which a car approaches a pedestrian crossing. How should a car approach the crossing when it does not have full information regarding its surroundings [5]. Should the car travel at a speed which at which it can always out maneuver any pedestrian that runs onto the street, but if it does then it will be going too slow and cause disservice to it's occupants. The car must make ethical decisions for mundane scenarios as well. This means instilling certain behaviors and biases in the car that might feel very natural to humans, but illogical and counter productive to an algorithm that is designed to minimize loss and maximize safety or efficiency. Thus an ethical car has to have the context of it's surroundings to make the right decision.
Although, this concept of mundane ethics seems very abstract and hard to implement into a computerized system and Johannes Himmelreich does not offer a solution to the presented problem, its importance cannot be overstated. Cars are huge physical objects that interact with the world in impactful ways. The importance of minute interactions and decisions cannot be over looked.

In a paper written by Levent Yilmaz, he underlines the importance of taking into consideration the cognitive perspective as well as characteristics of complexity in relation to verification and validation of autonomous systems. He goes over the challenges in decision making in complicated scenarios and outlines a method to make such decisions. By his definition, self-driving cars have to be either implicitly ethical agents (avoid any unethical scenarios), explicitly ethical agents (actively choose the most ethical decision) or a fully ethical agent (have the expertise to make the right decision, reason why it made that decision and learn from its errors). The type of agent the car is defined by the architecture and implementation of its ethics systems. Such an agent would have to operate in a non-linear open and dynamic world. The openness of the environment can lead to multiple courses of action that can have favorable outcomes. Yilmaz proposes a design to validate the right decision [6].

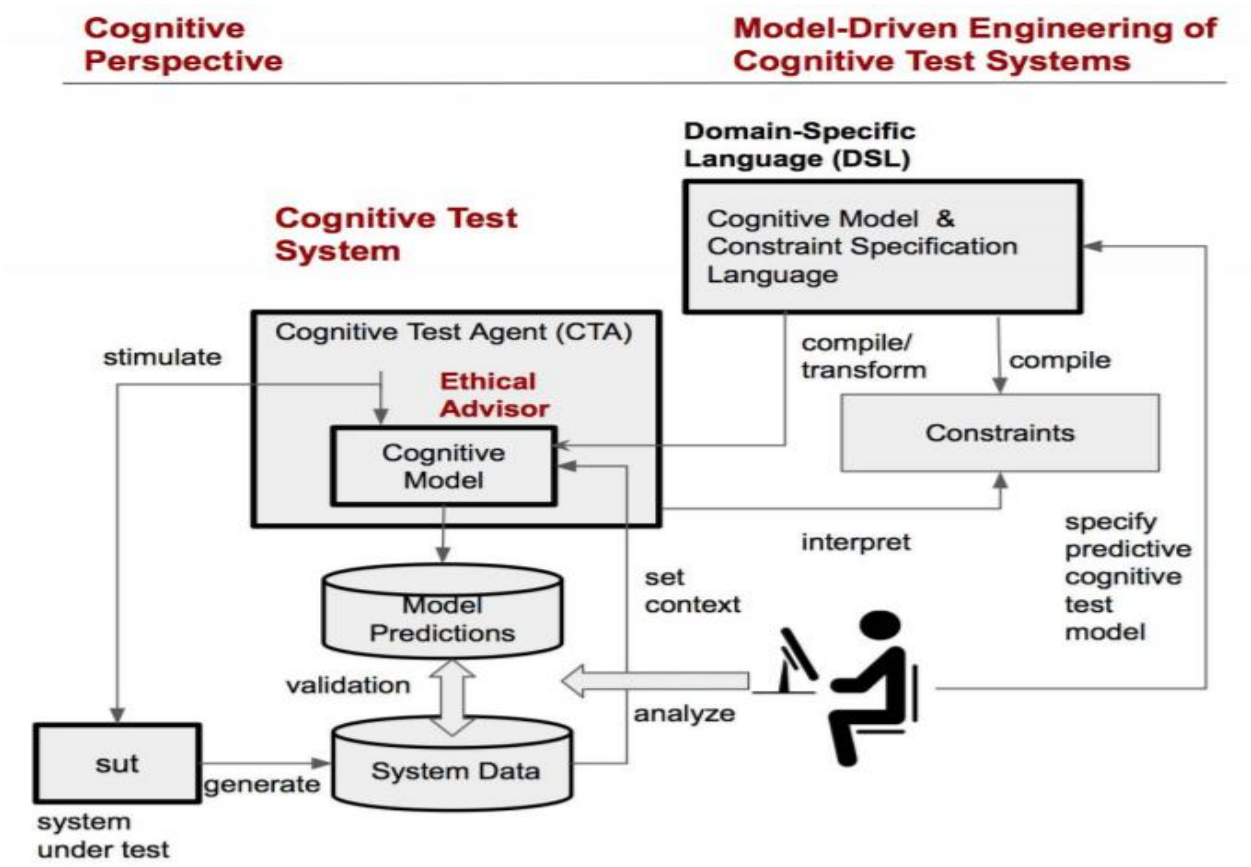

Fig 4.3: Cognitive System

The cognitive model takes in multiple layers of input to create an image of the state of the agent and the environment the agent is in. The cognitive model makes a prediction based on that input and then the system makes a decision based on that prediction. The cognitive model works by representing the environment and as constraints and trying to make a decision by solving a constraint satisfaction problem. Each constraint has a weight and the model tries to fulfill the most constraints while keeping their weights under consideration [7].
This approach requires a lot of data to work and a lot of sensory information to create a model that can make such a prediction. This type of data cannot be fed in raw to the system but has to be preprocessed and converted into constraints. This collection of data and pre-processing of it, adds much more complexity to the system. Even though the papers written by Yilmaz don't mention it, in order to make a predictive system, the system has to have a temporal memory and some already existing mapped understanding of it's environment, which raises the complexity even more. The ideas presented in the papers can still be of use, a simplified version of this system can be used to create an agent that runs on limited information.

There are military applications for such an ethical system. Such a system in theory could decide which Course Of Action (COA) to take under specific circumstances. In the paper "A principles-based model of ethical considerations in military decision making", it is discussed how a simple system that calculates

the most ethical of 2 possible COAs can be developed. The system relies on humans judging a predefined scenario and the data from that is used to decide which COA to take [8].

$$\Delta V_{AB} = \sum_{j=1}^{n_p} (v_{Bj} - v_{Aj}) w_j$$

The COAs (a and b) are translated to vector spaces and represented as vectors. In the equation above, Vb and Va are 2 courses of action, their respective weights are summed up and if the resulting value is < 0 Va should be considered, if it is > 0 Vb should be considered. The weights are predetermined values. Although this linear approach is too simplistic for a car interacting with it's environment (the decision to choose a COA available to a car is rarely binary). However, the paper does indicate that even though humans are good at identifying information that might be pertinent to the scenario at hand, they don't perform as well when they need to use the information to make a decision. This system leverages humans to identify the pertinent information in terms of weights for each COA and uses a mathematical model to calculate the preferred outcome. Translating possible COAs to vector spaces and calculating the preferred outcome could be useful tool in choosing the correct action for a self-driving car

Challenges in making an ethical self-driving car

There are certain practical and technical challenges which cannot be ignored when designing any sort of autonomous decision-making vehicle. Tobias Holstein, in the paper "Ethical and Social Aspects of Self-Driving Cars", outlines some of the requirements and challenges that arise when undertaking such a project [9].

One issue outlined is the tradeoff between the amount of data collected and processed and the processing time for the system. The system must run on already existing hardware, and hence has to perform well enough to be usable. The more data that needs processing, the more time the system sends processing it and the longer it takes for the system to act.

A related issue is the knowledge of self. Any entity that needs to make an ethical decision has to be aware of its own existence and the impact it can have on it's own environment. The system needs to be aware of the space it occupies in the world via a collection of sensors and algorithms. This simplistic self-awareness must be coded into the system.

Another issue that directly affects our project is the need for easy testing and standardization between systems. Autonomous cars cannot be tested the same way human drivers can. There needs to be a way that such cars can be tested cost effectively and that doesn't require a bespoke testing solution for every possible autonomous car on the market. The testing should not only judge whether the car made the right decision but also why it made that decision and whether that decision making is consistent across different scenarios. This transparency should not be at the cost of the intellectual rights of the company that made that car.

Building a multi-agent system

The scope of the problem is exceptionally large. It cannot be tackled by 1 system alone, and many of the cars functionalities must be delegated to smaller sub-systems which perform different tasks, the goals of each system are in service to the overall aim of the car: to navigate safely within the environment.

The sub-systems in the car may employ learning algorithms and may act as individual agents. As such

there might be emergent behavior between these sub-systems over time. We intend to review emergent systems and how to predict or model their behavior. One approach to modelling emergent behavior is “Agent-based modelling and simulation”.
The paper “Tutorial on agent-based modelling and simulation”, outlines the method above and shows how to implement In various scenarios in which the agents within a system interact with each other and the environment the system is placed in. The agents within a system operate within an “agent space” while being fed information about the environment the system is in. The agents can interact with each other within the “agent space” [10].

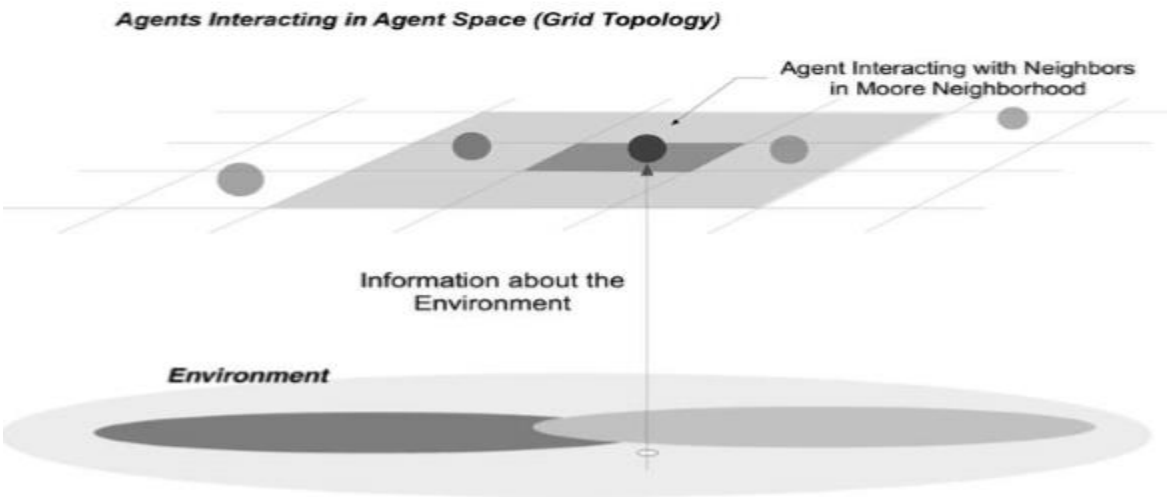

Fig 4.4: Agent Interaction in Agent Space

Such a method could be useful when designing a multi-agent self driving system. However, the paper only provides highly abstracted examples of implementing such a method.

Large Scale Visual Recognition Challenge (ILSVRC)

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) evaluates algorithms for object detection and I mage classification at large scale. One high level motivation is to allow researchers to compare progress in detection across a wider variety of objects -- taking advantage of the quite expensive labeling effort. Another motivation is to measure the progress of computer vision for large scale image indexing for retrieval and annotation [11].

ILSVRC has delivered some very promising and impactful advances in the field of computer vision and object detection. Some of the past winners have been AlexNet, InceptionV3 and ResNet. As detailed in the previous report, we are using a single image as our input to our system. Learning from these advances in this field and these algorithms will benefit us greatly (as we shall see below).

Alexnet

AlexNet famously won the 2012 ImageNet LSVRC-2012 competition by a large margin (15.3% VS 26.2% (second place) error rates). The main advances that AlexNet pioneered that are relevant to us are [12]:

1. Use of Relu instead of Tanh to add non-linearity. It accelerates the speed of training by a factor of 6 while still maintaining a respectable accuracy.
2. Use dropout instead of regularization to deal with overfitting [13]. However this has a training time penalty
3. Using overlapping pooling and convolutional layers to reduce the amount of parameters in the NN, decreasing the training time.
4. Using Local Normalization Response Layer to prevent over excitation of certain neurons in local neighborhoods, this also prevent over fitting. (this will be explored further as well)

AlexNet was trained over a dataset 1.2 million datapoints for 1000 classes. Testing AlexNet for use in one of our sub-systems should reveal some insight into how convolutional and pooling layers can be used to detect features from the provided image.

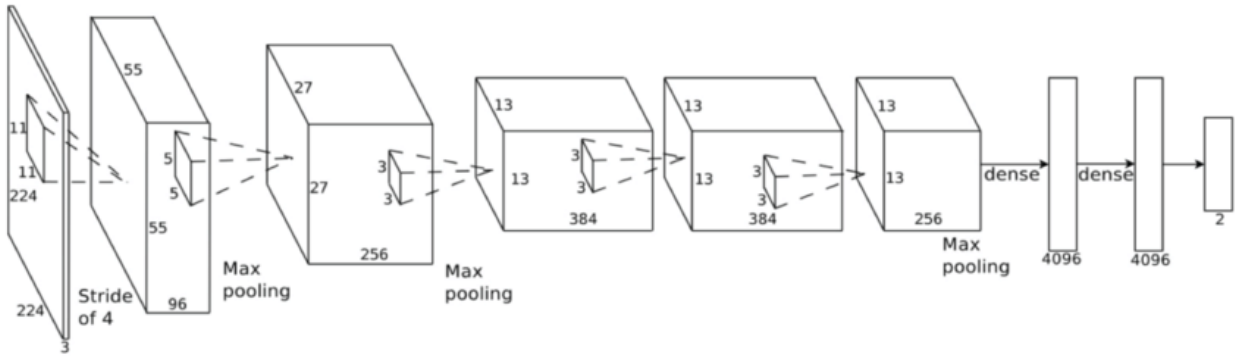

Fig 4.5: AlexNet Architecture

InceptionV3

The Inception network was an important milestone in the development of CNN classifiers. Prior to its inception (pun intended), most popular CNNs just stacked convolution layers deeper and deeper, hoping to get better performance.

The problem faced by conventional CNNs is that they are built to be deep. They have convolutional and pooling layers with fixed parameters(such as padding and window sizes). This can cause them to miss

out on some features that, if extracted, could result in better accuracy. InceptionV3 fixes this by going wide instead of deep. It employs multiple convolutional and pooling layers in parallel and then merges them to extract the maximum amount of features. This also has the affect of reducing the number of parameters in the NN [14].

InceptionV3 was trained over a dataset 1.2 million datapoints for 1000 classes.

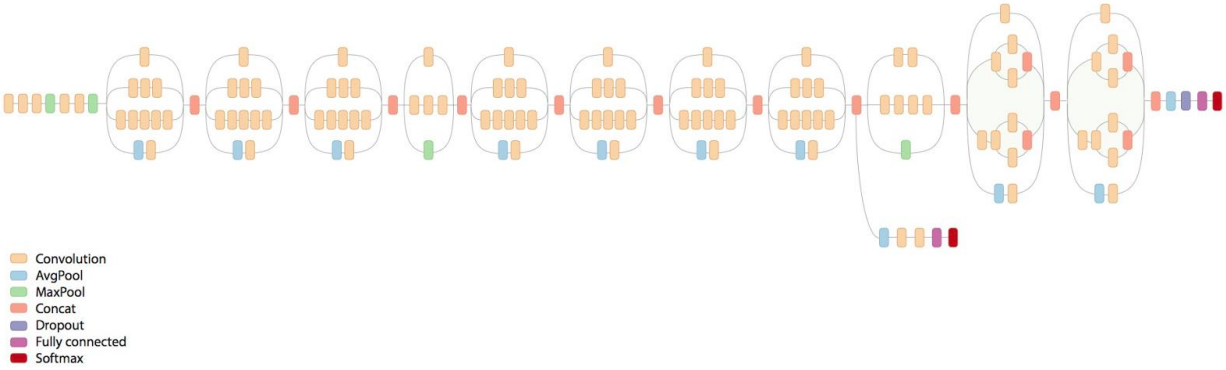

Fig 4.6: InceptionV3 architecture

We can use inceptionV3 to see how many relevant features we can extract from our image, and see if we can tune the architecture to exact precisely the features we want.

Local Normalization Response (LNR)

In neurobiology, there is a concept called “lateral inhibition”. Now what does that mean? This refers to the capacity of an excited neuron to subdue its neighbors. We basically want a significant peak so that we have a form of local maxima. This tends to create a contrast in that area, hence increasing the sensory perception. We want to have the same thing in our CNNs.

Local Response Normalization (LRN) layer implements the lateral inhibition we were talking about in the previous section. This layer is useful when we are dealing with ReLU neurons. ReLU neurons have unbounded activations and we need LRN to normalize that. We want to detect high frequency features with a large response. If we normalize around the local neighborhood of the excited neuron, it becomes even more sensitive as compared to its neighbors [15].

At the same time, it will dampen the responses that are uniformly large in any given local neighborhood. If all the values are large, then normalizing those values will diminish all of them. We want to encourage some kind of inhibition and boost the neurons with relatively larger activations. This has been discussed nicely in Section 3.3 of the original paper by Krizhevsky [11].

Our dataset contains of only images of the road from the perspective of the car’s hood. This means that most of the images from our dataset will look like this:

Fig 4.7: Typical screen capture from simulation environment

This means that there are a lot of neurons that will get activated by the same features which might cause some ReLU neurons to get high activation, and overshadow neighboring neurons. This will result in the NN failing to detect slight changes in the features of the image and this is something we want to avoid. LNR layers can help us deal with this specific issue.

Further Literature Review

For this period of our project development, we focused on how to prepare and pre-process the collected data.

Feature Normalization in Dataset

Feature extraction and feature normalization is an important preprocessing technique, usually employed before classification. Feature normalization is a useful step to restrict the values of all features within predetermined ranges. However, appropriate choice of normalization technique and normalization range is an important issue, since, applying normalization on the input could change the structure of data and thereby affecting the outcome of multivariate analysis and calibration used in data mining and pattern recognition problems [16].

For technical reasons, which will be discussed later in the report, the input to our system is a downscaled version of the screen capture of our simulation environment. The input is of the resolution 160x120 and has a single channel (black and white).

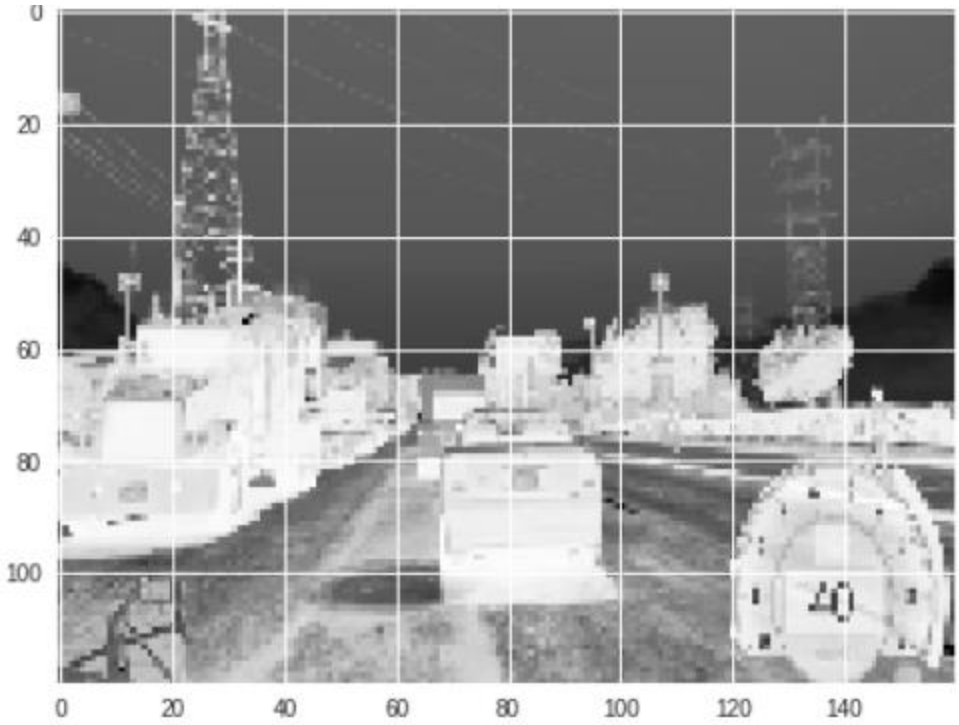

Fig 4.8: Input frame to system

The process of downgrading the resolution and transforming it from RGB to black and white causes some visual artifacts to appear. Some regions become very bright, and this leads to loss in detail. This can somewhat be remedied by performing image normalization. Each individual pixel is normalized over the whole image. This can be done through the following formula [17]:

(I_n: new_intensity) = ((I_o: old_intensity)- (I_o_min: old_minimum_intensity)) x ((I_n_max: new_maximum_intensity) - (I_n_min: new_minimum_intensity)) / ((I_o_max: old_maximum_intensity) - (I_o_min: old_minimum_intensity)) + (I_n_min: new_minimum_intensity)

Fig 4.9: Normalization done on sample image [17]

5. System Design

This section will focus on the final iteration of the system(V 0.2). The previous iterations of the project, their analysis and shortcomings have already been detailed in previous reports.

Also from now on the system will be referred to as CID, short for Can It Drive (pronounced as sid).

Architecture

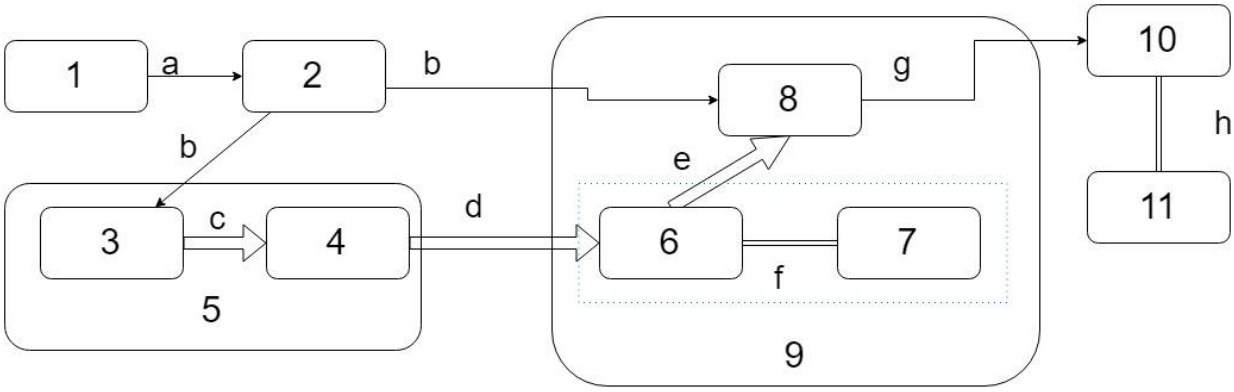

Fig 5.1: CID Architecture

Each Box indicates a sub system that can be developed and trained independently and the plugged into CID if the subsystem can deal with standardized input and outputs of the sub systems in CID. The dotted boundary around 6 and 7 indicate that even though they are independent systems, they should be developed together as they are closely linked to each other.

Solid arrows indicate image frame being passed

Hollow arrows indicate certain parameters being passed

Double solid lines indicate query and reply

Subsystems

1. Screen Image Capture
2. Image Filter
3. Object Detection Neural Network
4. Object Filter
5. Object Detection
6. Inference Engine
7. Knowledge Base
8. Image modifier
9. Ethics Engine
10. Navigator

11. Contextual awareness

Parameters passed between subsystems

a) Original image captured from simulated environment

b) Filtered image (image sent to different sub-systems are filtered differently depending on the sub-system)

c) Attributes of objects in the image (Such as: Object type, bounding box size, bounding box location)

d) Attributes of only relevant objects in the image

e) Modification command

f) Query to and reply from Knowledge base

g) Final modified image

h) Contextual awareness command

Life-cycle of a single input:

1. The image is captured and filtered according to the final destination of the image
2. Any objects in the image (such as traffic lights, cars, pedestrians) are detected, and the attributes of the relevant objects are passed on to the inference engine
3. The inference engine looks at the attributes passed to it and after consulting the knowledge base decides whether to influence the action taken by the navigator by modifying the input image to the navigator
    3.1. The inference engine can decide to leave the image completely unmodified
    3.2. The inference engine will have a library of modifications which corresponds to certain actions
4. The modification command is sent to the image modifier system, and it executes the command accordingly
5. The navigator takes the new modified image as input, and outputs the corresponding command to the car in the simulated environment
6. The contextual awareness system tells the navigator what environment it is driving in. The navigator will use the correct model to then act based on the input. (For example: If the weather changes from sunny to snowy, the contextual awareness system will notify the navigator to use the model that was trained to drive in snowy environments)

The major improvement between this architecture and the previous one, is that the newer version gives clear roles and controls to the different sub-systems. This creates a defined lifecycle for the input that we can use to implement our sub-systems and collect data to train those sub-systems.

Previously the different sub-systems were all getting input images and then outputting an action to a control sub-system which would then choose which sub-systems output (based on a hierarchy) should be forwarded to the car. This produced additional complexity and needless redundancy in the system as the roles of the sub-systems overlapped with each other.

As stated in the first report, for an autonomous car to be ethical, it needs to overcome mundane ethical solutions. It also needs to have explicit and implicit awareness of the context of the surroundings. We plan to achieve this in much the same way as described in the previous report.

CID will be given context explicitly through the contextual awareness sub-system, and implicitly via the life-cycle of the input to the Navigator system. The ethics engine will provide explicit commands to the car on what not to do. For no we are aiming to program the car as an implicitly ethical agent [6]. CID will only make decisions to avoid unethical situations by reaction, rather than predicting situations and doing the explicit action before the event can occur.

The parameters passed between the sub-systems and the final output from CID will be time stamped and logging into a log file on a per frame basis for later analysis.

Object Detection System

The Neural Network used in this system is a TensorFlow Model that is open source and publicly available [18]. The model is Faster RCNN, and it is trained over the COCO Dataset [19]. This model is trained to detect normal everyday objects in an image (cars, people, dogs etc).

We can use transfer learning to train the model on our own prepared dataset so the model can detect more objects. For now the model works good enough for the current iteration (CID V0.2).

Fig 5.2: Overlay of Object Detection System on grabbed image

The model creates a bounding box on every object it recognizes. It sorts the objects into classes based on the COCO dataset and gives each detected object a confidence rating. Based on the object class, the relevant objects' features will be extracted and sent to the Ethics Engine. The attributes that are passed to the Ethics Engine are:

1. Object class
2. Confidence level
3. Bounding box location
4. Bounding box size

Inference Engine

The inference Engine within the Ethics Engine ultimately decides what the input to the Navigator should be. The navigator now is the only sub-system that outputs directly to the car.

The inference engine does not see the image itself in any form. It only receives input in the form of attributes of relevant objects in the system. The inference engine must make decisions on what the navigation should do based on the current image. The decision is based on the objects in the image that the car can interact with. Hence, the only features relevant to the inference engine are the attributes of the objects in the image, and not the whole image itself.

Passing only the attributes of objects to the inference engine gives us flexibility in implementing the inference engine. We can either implement a neural network to predict the right action to choose based on the attributes of the object. Or we can use and "if else" algorithm set hardpoints for the values of the attributes passed to the inference engine (for example: if object is traffic light, if object size > x and location = y then do something).

Knowledge Base

The Knowledge Base (KB) includes the details of the objects that CID can interact with, and a library of commands that the Ethics Engine can pass to the navigator. We will discuss these commands later.

Semantics Graph

The semantics graph is a way in which we can detail the classes of objects that CID can interact with. The graph is a binary tree, in which each leaf has a "is a" relationship with root node. Object classes are segmented into binary categories at each node.

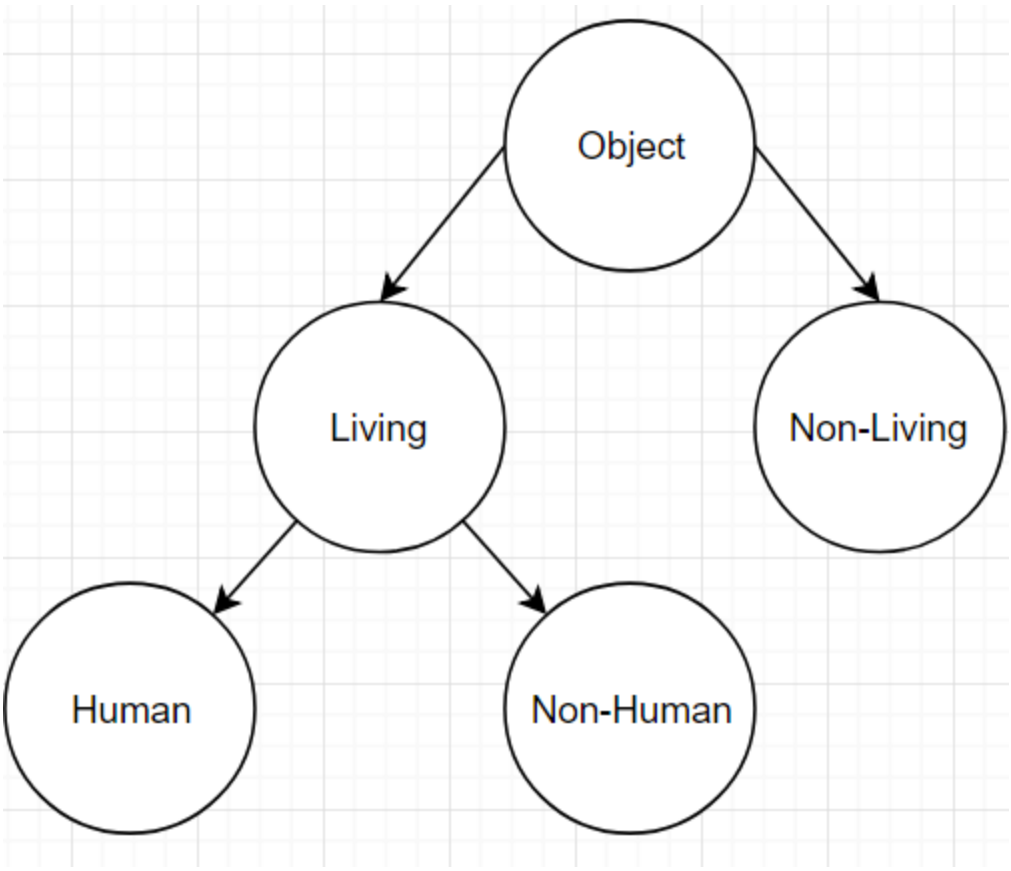

Fig 5.4: Example Semantics Graph

Each Sub-tree has a priority modifier, that applies to all the nodes in the tree. I.E, if the priority modifier of living is 3 and Non-Living is 1, then any living object will always be given 3 times the priority as given to the non-living object. The same can then be repeated for human and non-human. If human has a modifier value of 2 and non-human 1, and a boy is classified as a human and a dog a non human, the boy will always be given twice the priority as compared to the dog.

This Semantics graph makes it easy to classify and allocate attributes to the objects that CID might interact with or encounter while driving. In CID V0.2, the semantics graph has the classes that the object detection system can detect (objects in COCO dataset).

Navigator and Contextual Awareness System

The navigator system is trained to drive the way a responsible human driver would drive. The navigator system is designed to run from multiple models, each trained to drive in different circumstances and environments (city, highway, sunny weather, snowstorm etc).

The Contextual Awareness system tells the navigator what the situation CID currently is in, and the navigator gets the behavior from the relevant model.

Behavior Modifiers

The most crucial aspect of CID's architecture is how the Ethics Engine and the navigator communicate with each other. The navigator has to work independently on it's own even when it gets no input from the ethics engine, and when it does get the input from the ethics engine, the navigator has to react in a natural and appropriate manner.

This means that when the ethics engine tells the navigator to stop, the navigator should not just apply the breaks and come to a complete halt all of a sudden. Instead it should stop taking into consideration the current frame that the car is in, making sure that it does not collide with any other object when coming to a stop.

This means that the navigator has to preserve it's original behavior, while at the same time be able to adapt to the inputs from ethics engine. We wanted to preserve CID's modularity and ease of development, so instead of retooling the ethics engine and the navigator to fulfill these objectives, we came up with a way to allow the 2 sub-systems to communicate and be trained in such a way that CID can behave as desired.

Our approach to solve this problem is to allow the ethics engine to modify the image by adding certain features to the image, that the navigator can detect. The navigator will be trained to react to these new features accordingly. The knowledge base will hold a library of these modifiers which the ethics engine can access. Each modifier feature will correspond to a certain action (For example: feature a means to brake, feature b means to turn right).

These new features will not be superimposed onto the image, instead a new channel will be created for each image, and this new channel will only hold the behavioral modifying features. If the system is trained to run on black and white images (single channel), then the final image given to the navigator will have 2 channels. If the ethics engine does not choose to modify the behavior of the navigator, then this new channel will be left blank.

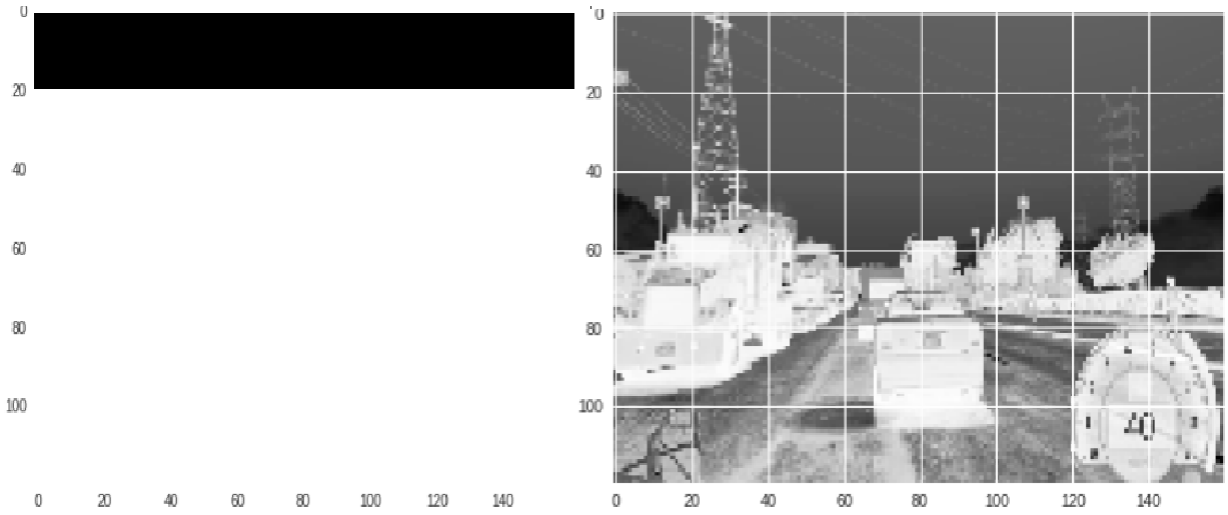

Fig 5.5: Sample input with original channel and new modifier channel

The image above shows a sample input to the navigator system. The feature added to the second channel is a black bar at the top that is 20 pixels thick. The ethics engine is telling the navigator to stop,

the navigator system is trained to apply brakes whenever it sees this feature in the second channel, it takes in the features from the original channel as well and stops accordingly.

Another benefit of this approach is that the ethics engine can communicate more complex commands to the navigator. In the second channel, each pixel can have a value between 0 and 255. The ethics engine can activate each individual pixel to a value between 0 and 255 to create a new feature. By modulating the value of each pixel, the ethics engine can then convey the urgency of the command. If the feature has a high value, the command from the ethics engine is urgent, and vice versa if it is not urgent.

Furthermore, the ethics engine already knows the objects, their location, size and priority that are present in the input frame. It knows this because of the inputs from the Object Detection system and it's own knowledge base. The ethics engine can then add features to the second channel that highlight which objects to avoid in the input and the priority of each object. The navigator system must be trained accordingly in order for it to understand all these commands.

Essentially, these behavioral modifiers allows us to change the behavior of CID at real time in a very nuanced way.

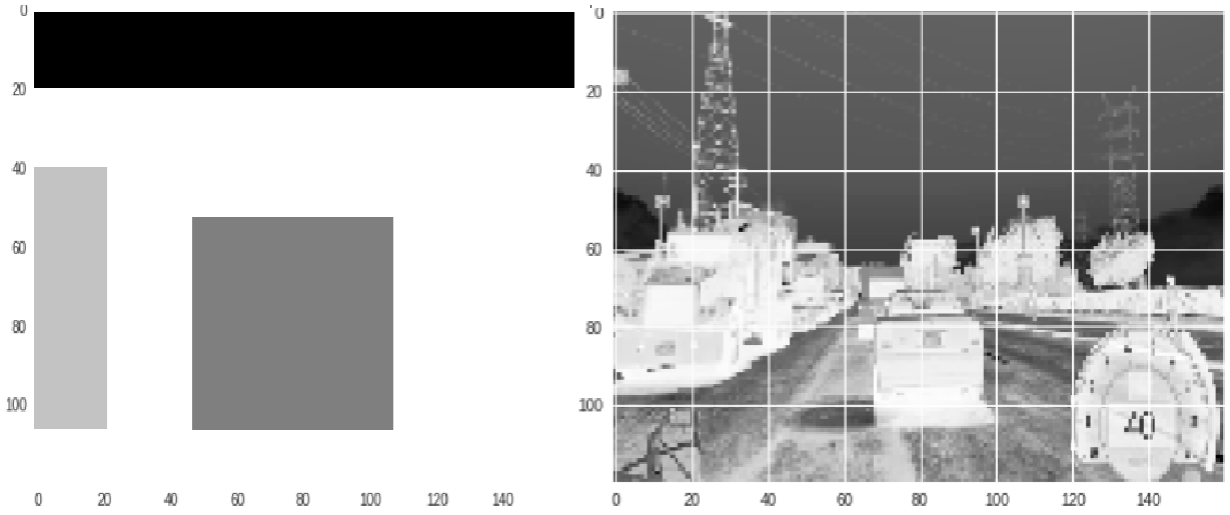

Fig 5.6: Sample input with original channel and new modifier channel with additional features

In the sample input above, the bar at the top is black, which indicates a higher pixel value, which signifies the urgency of the command related to that feature. The Object detection system detects a car in the middle of the road and a semi-truck on the left-hand corner. If the ethics engine wants to communicate with the navigator to avoid these 2 objects and the priority values of these 2 objects, it can do so by creating 2 new features in the second channel which correspond to their locations in the frame. The car in the front is closer to CID hence it gets higher priority than the semi-truck, and that's why its pixels (dark gray) are higher value than the pixels of the semi-truck (light gray).

This theory for modifying the behavior of a neural network while maintaining the old behavior was first tested on a sample dataset and a sample neural network. A neural network was trained to identify hand written digits. It was then trained to add 1 to it's prediction if a certain feature was added to the secondary channel. This model was demoed to Mr. Enver and Mr. Okan.

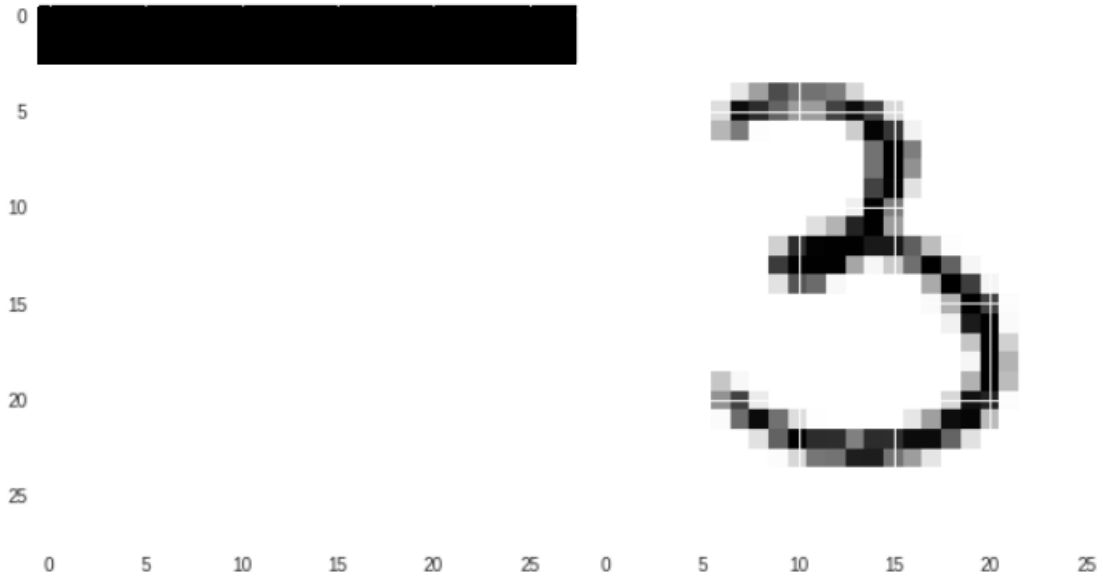

Fig 5.7: Input to test model

The output of the model to the input above is 3+1 = 4. If the model was given an input with the same image of 3 but the secondary channel was left empty, the output is 3.

The test model used the same type of convolutional neural network that was used in the navigator system.

6. Implementation

The milestones set for this semester were:

1. Have basic a workable demo of CID
2. Implement object detection and navigator sub-system
3. All sub-systems should be able to communicate with each other

We intentionally left out the implementation of the ethics engine, as we wanted to focus on defining an approach which would make it easier for the implementation of an ethics engine.

We have already outlined how all the sub-systems will communicate with each other. The Object Detection system is a pretrained model that we modified to run with our system. We only had to edit the code of this pretrained model to get frames from the simulation environment, and produce outputs in a way that the ethics engine can parse. We did not train this model.

As detailed in the previous report, we wanted to implement preexisting convolutional neural networks (Such as AlexNet and InceptionV3) for the navigator sub-system. However, this was proved not to be a very useful approach as the dataset that these models were trained over is very different from our dataset. These models were trained to identify images of 1000 different classes (cats, flowers, dogs, bicycle, etc). The dataset in our model is of the image of the road from the view of the bonnet of the car, which means that there is very little variation from image to image.

In this section we will detail the iterations of the navigator system and the dataset that it was trained over and the results of the training. The data for this training session was recorded in the city at midday (12:00 pm) on a sunny day, with the presence of traffic and pedestrains. The navigator sub-system will be trained to react to the brake command as detailed in Fig 4.5. All the iterations were trained for 30 epochs, and 10% of the dataset was reserved in each iteration for validation purposes.

NOTE: A term that will be referred to in the following sections is over fitting. It occurs when the model starts memorizing the training dataset rather than generalizing the trends in the dataset. Over fitting can be detected when the validation accuracy plateaus as the training accuracy continues to go up.

Original Dataset

540,000 datapoints were collected for the following training sessions. Each frame in the dataset was down sampled from 1280x720 to 160x120 and from RGB to Black and White. This downsampling was down so that the model would train and faster and we could iterate on the models faster.

After down sampling the images, each image had an extra channel to it. If the corresponding label to that image was ‘s’, ‘sa’ or ‘sd’, then that image was duplicated, and the 30 pixel thick black bar was added onto the top of the 2nd channel of the duplicated image. (See Fig 4.5).

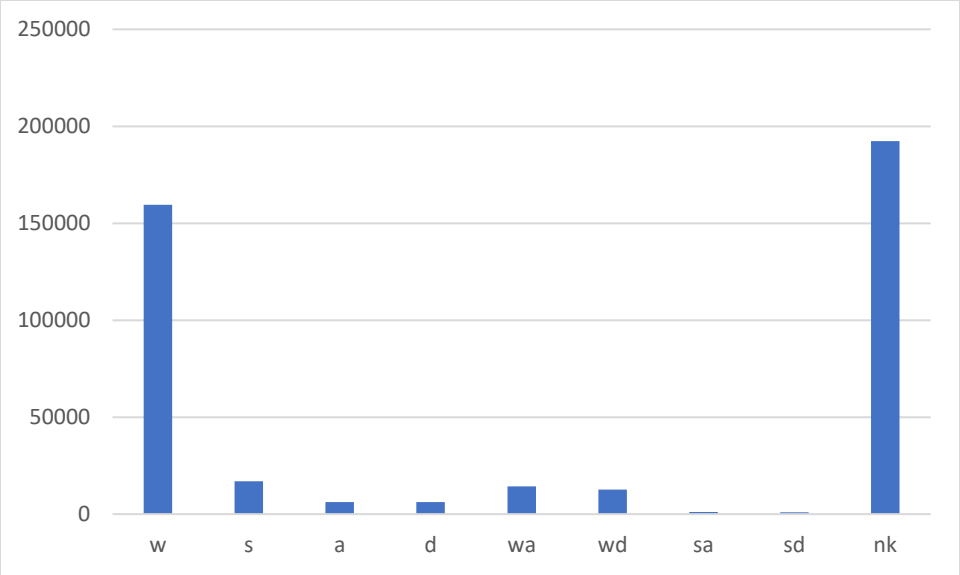

Fig 6.1: Distribution of datapoints

Navigator V0.20

Model

The model that this dataset was trained on AlexNet with the following modifications:

- The convolution and pooling window sized were reduced to deal with lower image resolution
- The model was modified from 3 channel input to dual channel input
- LNR (local normalization response) layers were added to deal with persistent high neuron activation to prevent overfitting

Dataset

For this iteration of the navigator, 6000 datapoints from each label were randomly sampled, and for label ‘sa’ and ‘sd’ datapoints were duplicated to get 6000 datapoints. This meant that a total of 54,000 datapoints (10% of original dataset) were used.

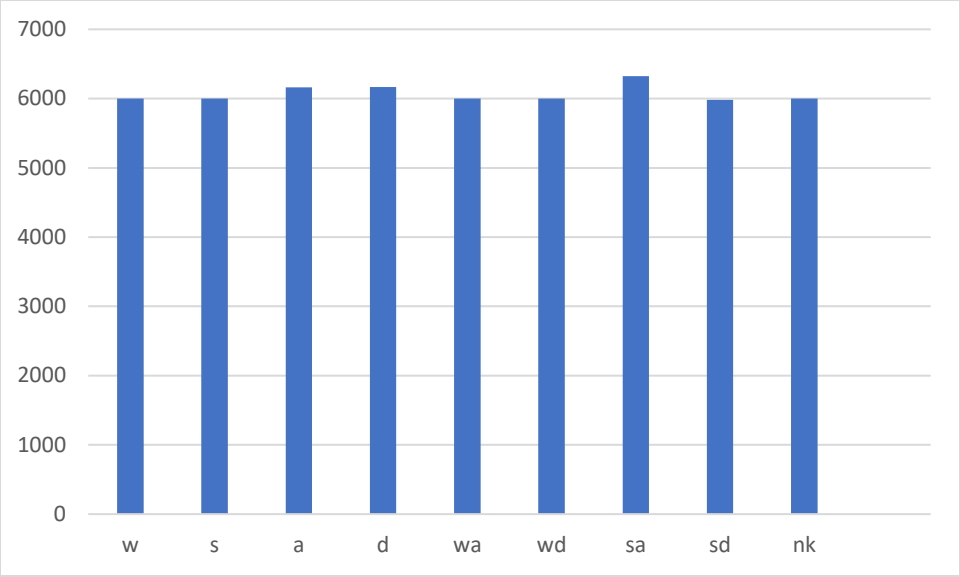

Fig 6.2: Distribution of datapoints for V0.20

Results

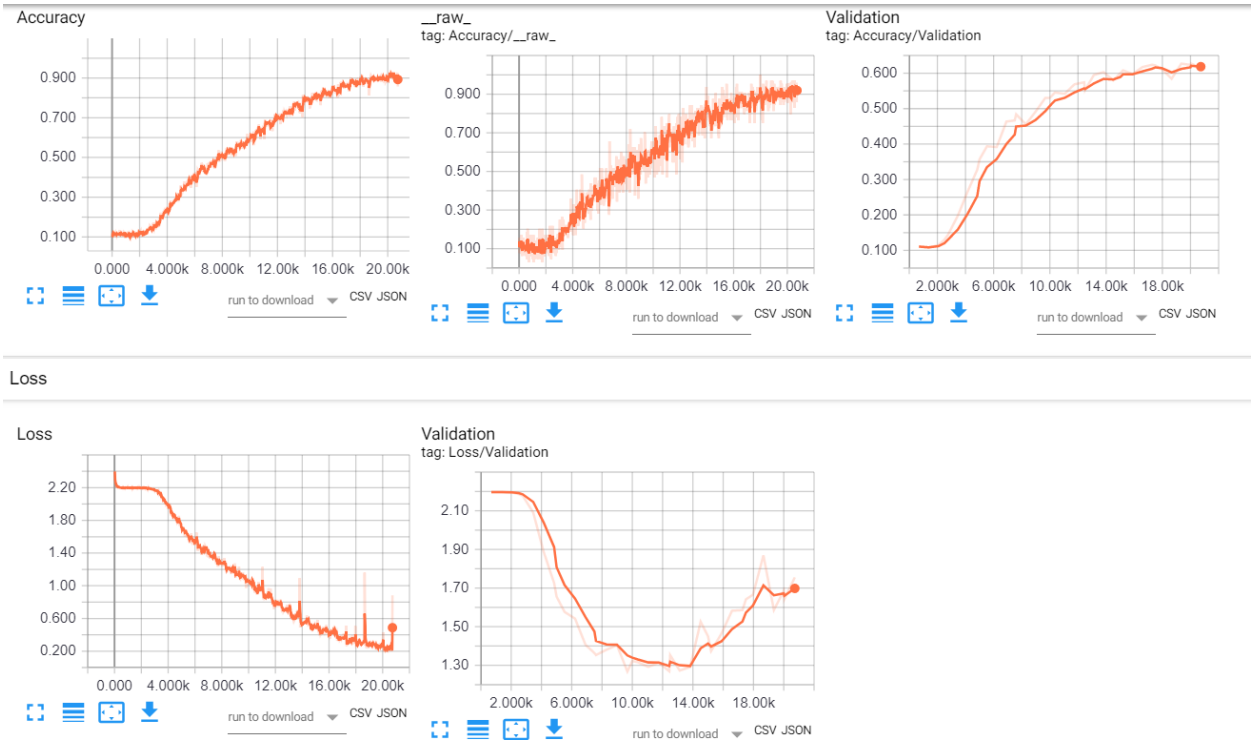

Fig 6.3: Results of V0.20

There is a clear difference in validation accuracy and training accuracy. Validation accuracy plateaus at around 60% while training accuracy goes upto 92%. The validation loss also has big spike at the end (we want to minimize loss). This signifies that the model was overfitting.

An assumption was made that the overfitting was occurring due to some of the data being duplicated for the purposes of balancing (specifically datapoints with label ‘sa’ and ‘sd’)

Navigator V0.21

Model

Model was unchanged

Dataset

For this iteration ‘sa’ and ‘sd’ were removed and the model was altered to work without these actions as possible outputs.

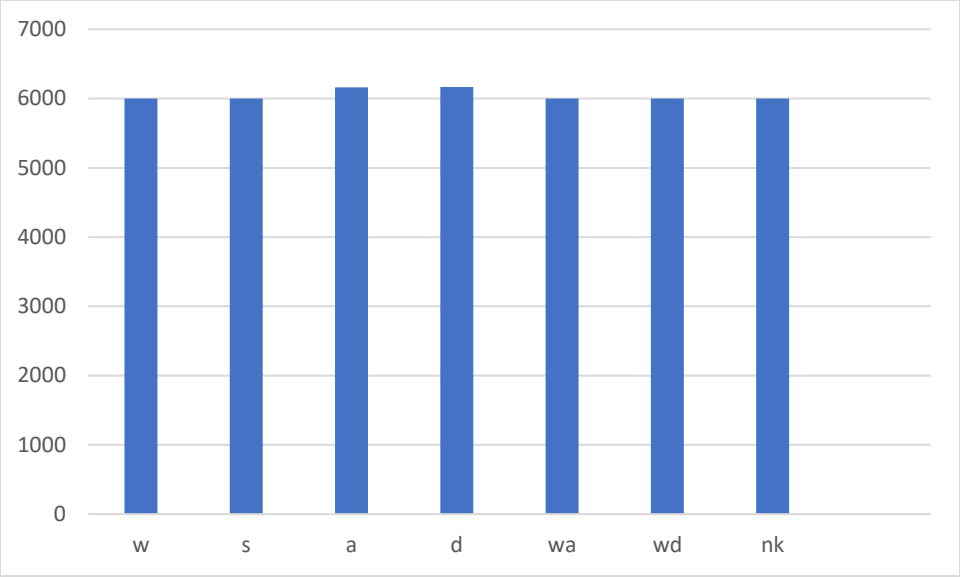

Fig 6.4: Dataset distribution for v0.21

Results

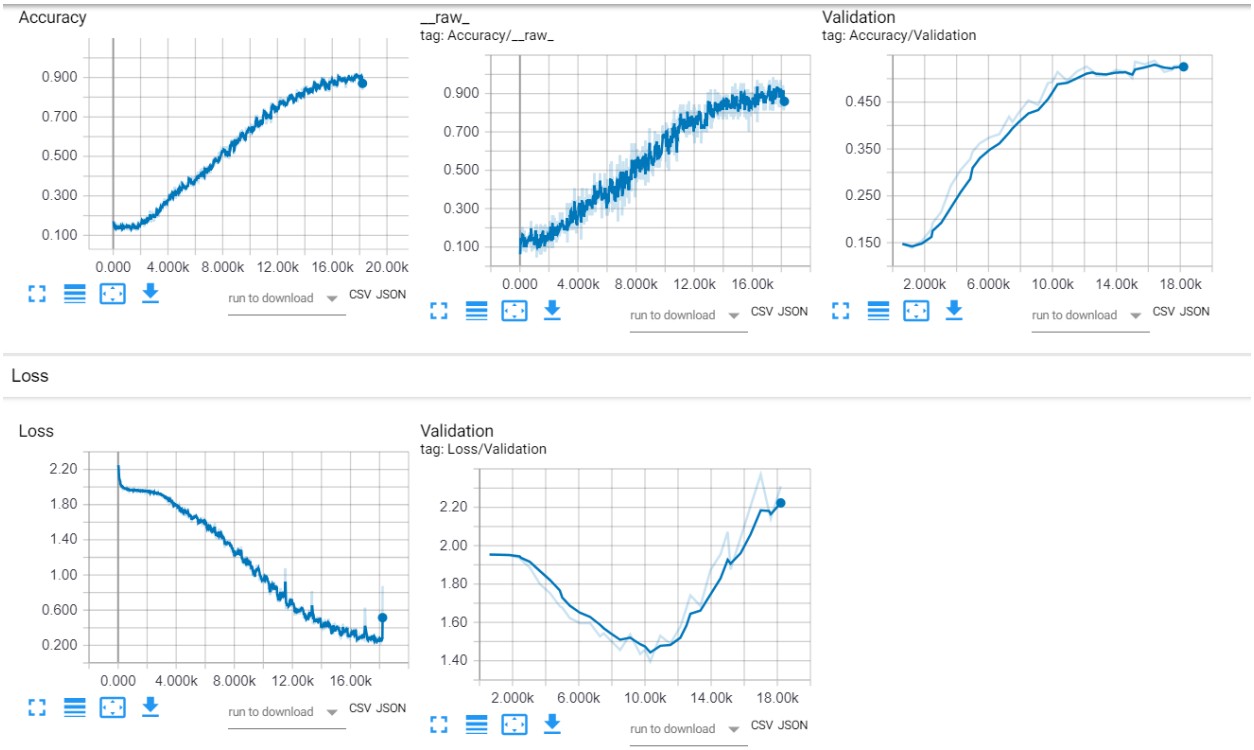

Fig 6.5: Results after training for V0.21

The model was still overfitting, even more aggressively now. There is a need to alter the architecture of the model to prevent it from overfitting

NOTE: V0.22 was lost due to a crash while training

Navigator V0.23

Model

The model was modified in the following way:

- Increasing Dropout layer rate from 0.5 to 0.7
- Removing 2 fully connected layers (4 fully connected layers before, 2 now)

These changes were made under the assumption that their were neurons that were getting over stimulated in the fully connected layer, and their were neurons that were getting under simulated in the fully connected layers. By removing 2 fully connected layers and increasing the drop out rate, we are forcing the model to reduce the over dependency between the neurons. This curbs the power of an individual neuron, and causes the model to focus more on the overall trend in the data, preventing over fitting.

The model was modified to work with only 4 possible actions (w, s, a, d)

Dataset

The dataset was further simplified. Only 4 data labels were used (w, a, s, d). 6000 datapoints from each data label were randomly sampled. Total datapoints used: 24000 (4% of original dataset)

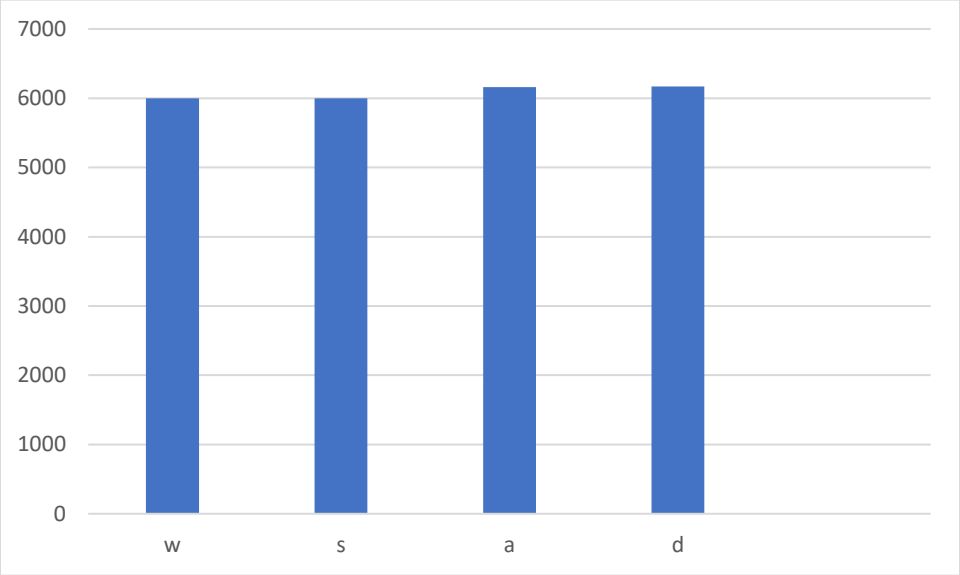

Fig 6.6: Dataset distribution for V0.23

Results

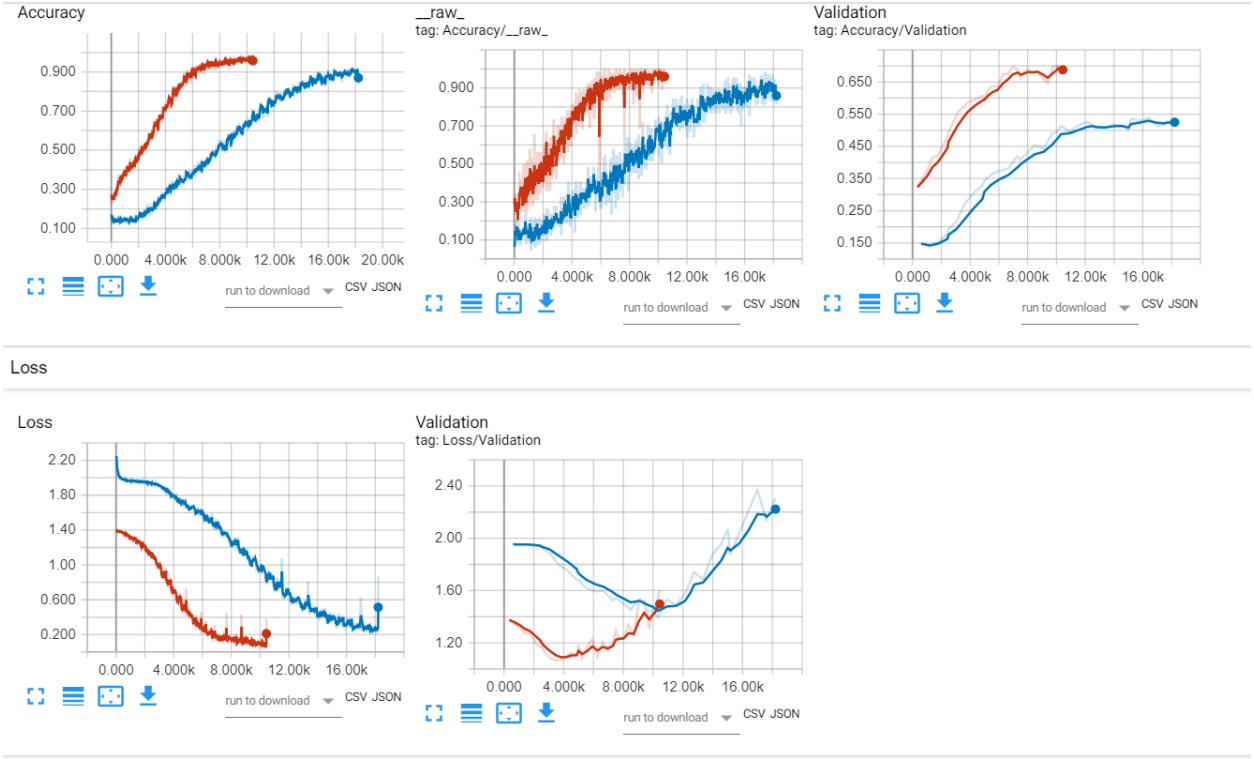

Fig 6.7: Results after training for V0.21 vs V0.23

V0.23 is in brown and V0.21 is in blue. There is a large jump in validation accuracy even though training accuracy remains the same. This jump in validation accuracy can be attributed to the fact that there are less possible actions for the model to take, which increases the chance of making the correct decision. The validation loss of the V0.23 however shows that the model is performing better, even when the increased probability of making the right decision is taken into account.

Navigator V0.24

Model

Modifications made to the model:

- Increased dropout rate from 0.7 to 0.85
- Decrease the number of fully connected layers to 1

Dataset

Dataset remains the same as in V0.23

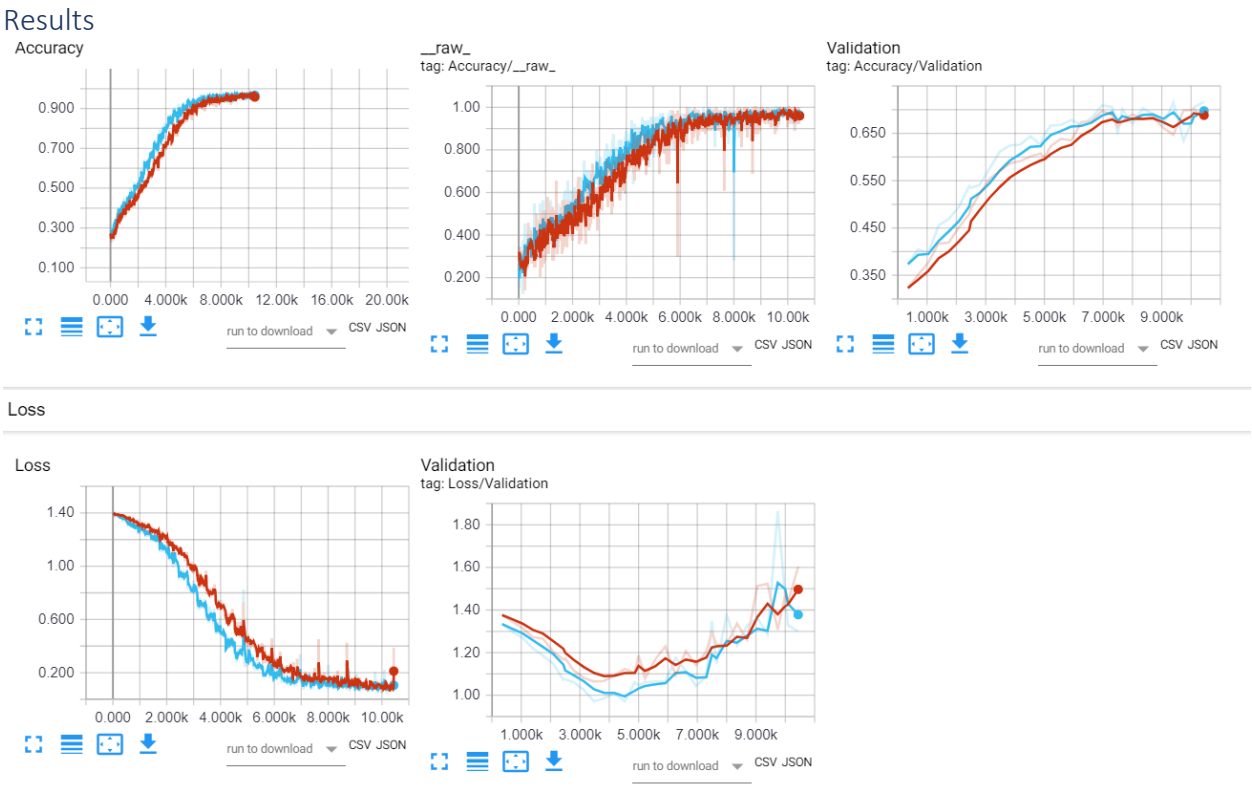

Fig 6.8: Results after training for V0.24 vs V0.24

V0.23 is in brown, V0.24 is in cyan. The validation accuracy and the validation loss have slightly better trends when compared to V0.23. This means that the model will not benefit any more from changing the dropout and fully connected layers any further

Navigator V0.25

Model

The model was modified in the following manner:

- Pooling layers were removed from the model altogether

Removing the pooling layer increases the number of parameters in the model drastically. This makes the model consider all the features in the image, rather than averaging the features of the image over a certain pooling window. The assumption is that this makes the model consider every small change in image, and each small change affects the decision made by the model. This change should increase the efficacy of our model by a large amount, as our dataset contains images which look very similar to each other, and have very small differences from 1 image to the next.

Dataset

Dataset remains the same as in V0.23

Result

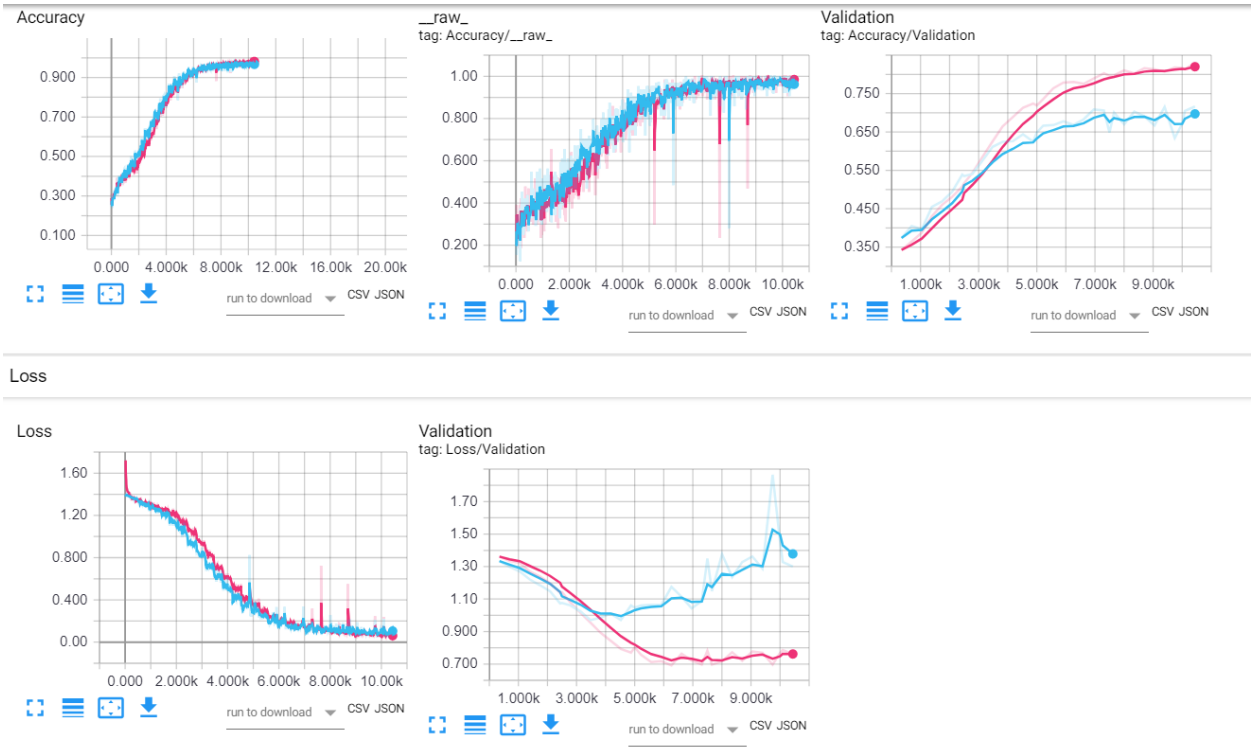

Fig 6.9: Results after training for V0.25 vs V0.24

V0.24 is in cyan, V0.25 is in pink. The results are immediately better. Validation accuracy shoots up to 81% and validation loss does not experience the same spike as seen in the other versions. The model is still over fitting slightly as training accuracy is 95% compared to the validation accuracy of 81%.

The downside of this change is that training time increased by more than a factor of 5.

Navigator V0.26

Model

Model remains the same as V0.25, it is only modified to take 5 possible actions now (w, a, s, d, nk)

Dataset

To prevent the slight overfitting seen, the amount of data is increased by a large amount. The data points for data labels ‘nk’ was added to the dataset. Data labels for ‘sa’ and ‘sd’ were added to data label ‘s’. Data labels for ‘wa’ were added to data label ‘a’, data labels for ‘wd’ were added to data label ‘d’. This was done to test how the model would deal with increased complexity.

18,500 data points from each data label were randomly sampled. A total of 92,500 datapoints were used for training (17% of the original dataset).

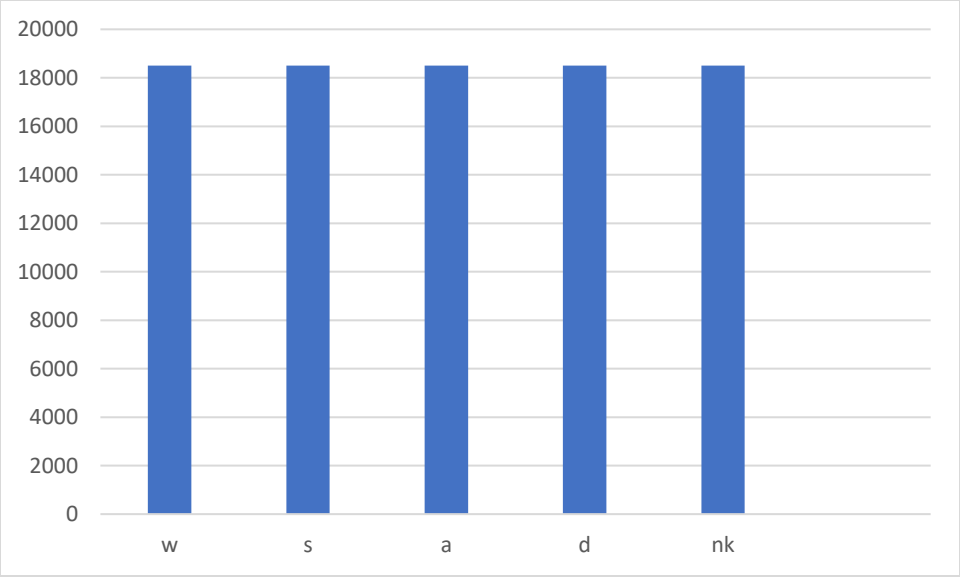

Fig 6.10: Dataset distribution for V0.26

Results

Because of the issues with training models on the cloud (google colab), this training session either kept timing out or crashing, no data or a trained model could be retrieved. These issues and their cause is detailed in report 2.

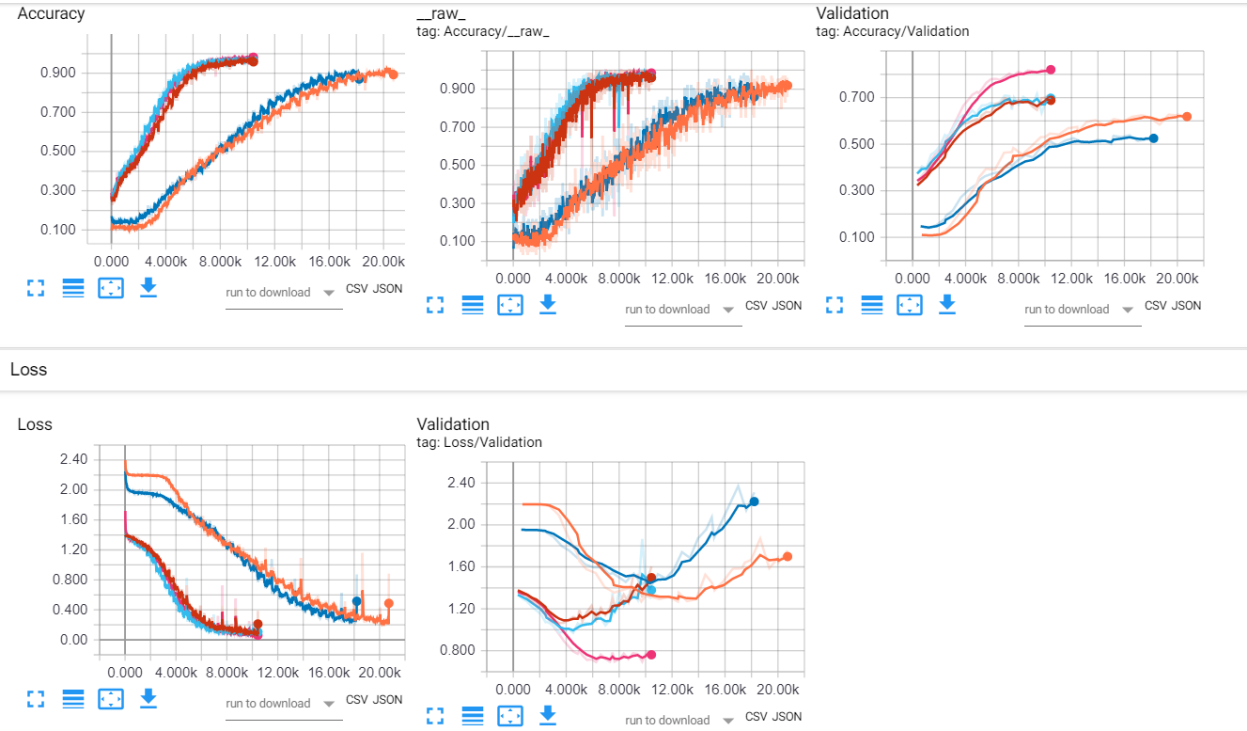

Fig 6.11: Training results of all iterations of navigator for CID version 0.2

V0.20: Orange

V0.21 : Blue

V0.23: Brown

V0.24: Cyan

V0.25: Pink

Deployment

The Navigator, Object Detection, and Ethics Engine Sub-system were all deployed independently and tested.

All these major sub-systems run on the graphics card. The neural network models in the Object detection and Navigator sub-system run off the graphics card through CUDA. The game itself runs on the graphics card.

The behavioral modifier is added to the input image by copying the screen grabbed frame from the game onto a dual channel template. Each pixel is copied 1 by 1 on to the 1st channel. The ethics engine then activates the relevant pixels in the 2nd channel 1 by 1 as well. Done on the CPU this process take O(n^4) time (n*n*n*n). This process can be parallelized on the GPU, and will then take constant time.

With the hardware available to us, the graphics card cannot handle all these sub-systems running at the same time, and it crashes. Hence, CID as a whole cannot be tested. At most, 1 sub-system, running alongside the simulation environment can be tested. However, even running just 1 sub-system can cause the PC to crash. CID does not run stably, even when only a part of it is being tested.

Fig 6.12: GPU performance when testing navigator sub-system

The image above shows the simulation environment being run alongside the navigator sub-system. The usage of the GPU immediately shoots up to maximum capacity. The navigator and the environment, both run very slowly. Eventually, the system itself crashes due to lack of memory availability.

This test was run on one of our laptops, with 1050Ti GPU. The PCs available to us in the lab have even less powerful Quadro k600 GPUs.

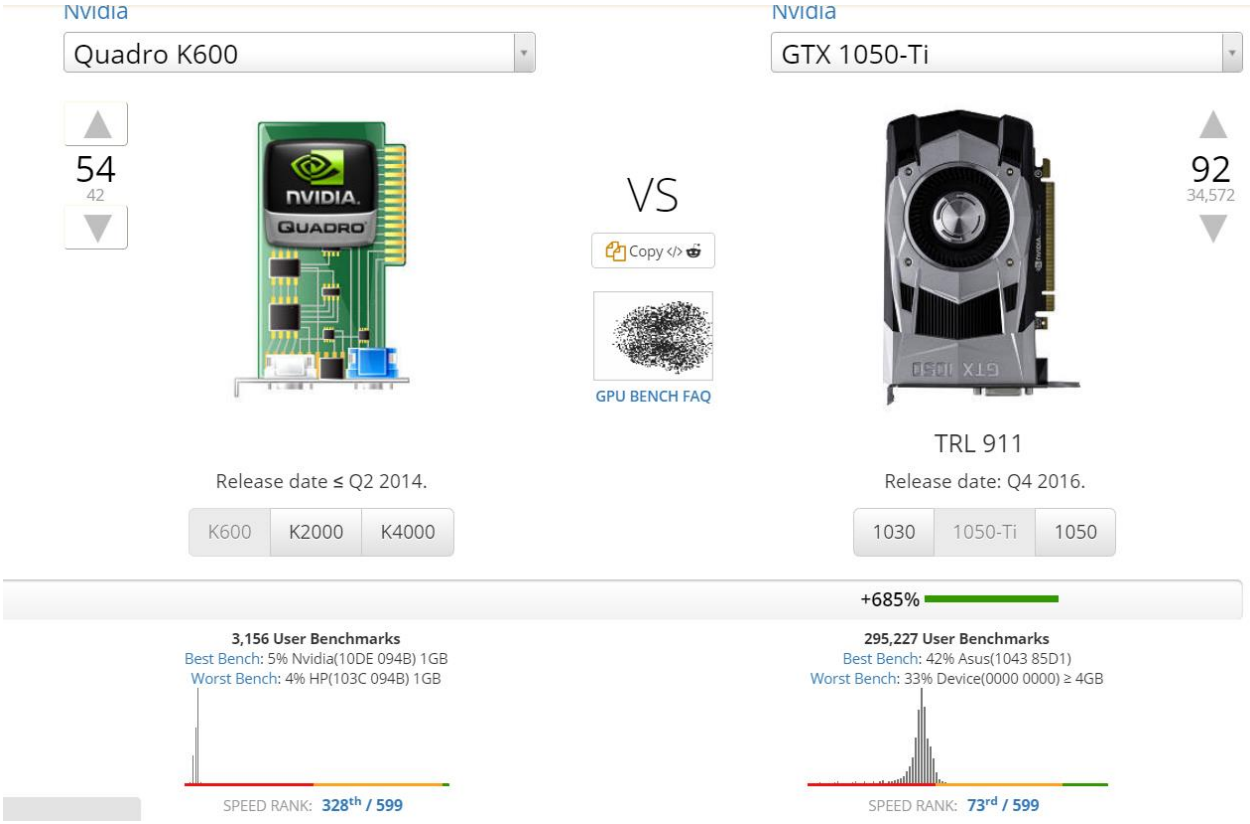

Fig 6.13: Performance comparison K600 vs 1050Ti

7. System Management and Project Estimation

Project Estimation

Function Point Estimation

General System Characteristic		Brief Description
1.	Data communications	How many communication facilities are there to aid in the transfer or exchange of information with the application or system?
2.	Distributed data processing	How are distributed data and processing functions handled?
3.	Performance	Was response time or throughput required by the user?
4.	Heavily used configuration	How heavily used is the current hardware platform where the application will be executed?
5.	Transaction rate	How frequently are transactions executed daily, weekly, monthly, etc.?
6.	On-Line data entry	What percentage of the information is entered On-Line?
7.	End-user efficiency	Was the application designed for end-user efficiency?
8.	On-Line update	How many ILF's are updated by On-Line transaction?
9.	Complex processing	Does the application have extensive logical or mathematical processing?
10.	Reusability	Was the application developed to meet one or many user's needs?
11.	Installation ease	How difficult is conversion and installation?
12.	Operational ease	How effective and/or automated are start-up, back-up, and recovery procedures?
13.	Multiple sites	Was the application specifically designed, developed, and supported to be installed at multiple sites for multiple organizations?
14.	Facilitate change	Was the application specifically designed, developed, and supported to facilitate change?

Fig 7.1: General System Characteristics

General System Characteristic	Influence(1-5)
1	0
2	3
3	5
4	2.5
5	5
6	0
7	4
8	0
9	5
10	5
11	2.5
12	0
13	0
14	5

Fig 7.2: Influence Multiplier

Influence Multiplier = 0.65 + [(Ci) / 100]
Where: Ci = degree of influence for each General System Characteristic

Therefore:
0.65 + [37 / 100] = 1.02

Function Points			
Program Characteristic	*Low Complexity*	*Medium Complexity*	*High Complexity*
Number of inputs	*0*	*1*	*1*
Number of outputs	*0*	*0*	*11*
Inquiries	*0*	*0*	*1*
Logical Internal Files	*0*	*0*	*1*
External Interface Files	*0*	*1*	*2*
Function Point Estimation			
Unadjusted total of function points	*18*		
Influence multiplier	*1.02*		
Adjusted total of function points	*18.36*		

Fig 7.3: Function Point Estimation Table

LOC Size Metric

We are using Keas (or TensorFlow Learn) over the TensorFlow platform in Python.

LOC per module/subsystem = 720

KLOC (KDSI) = 720 * 6 + 2000 = 6.320 Kilo Lines Of Code

Note: In the final iteration we did not add any more code so it is equal to the previous iteration

Our Basic COCOMO model is obviously semi-detached.

Hence:

a = 3.0
b = 1.12
c = 0.35
MM (Man-Month) = a * (KDSI)^b
MM = 23.65

Since it is semi-detached, TDEV (Total Development Time) = 2.5 * E^(0.35)

So TDEV = 2.5 * (23.65)^(0.35) = 7.56 calendar months

Project Tasks and their durations

Mahmoud's tasks

Task	Duration(hours)
Downloading and set up of simulation environments	10
Setting up modding tools	6
1st iteration of ethics engine	9
Data points collection	12
Meeting with supervisor	8
Developing an expert system prototype using Java with the help of IntelliJ IDEA	0.5
Testing viability of Google ML Platform	2

Fig 7.4: Sprint Table 1

Task	Duration(hours)
Researching appropriate cloud solutions for data storage	10
Setting up Google Bucket	2
Uploading data set to Google cloud	30
Reinstalling GTA V and mode after malfunction	6
Meeting with supervisor	8
Literature review	8
Data points collection	12
Researching appropriate cloud solutions for training	24
Uploading data points to cloud solution	10
Processing data points	12

Fig 7.5: Sprint Table 2

Task	Duration(hours)
Data points collection	10
Meeting with supervisor	8
Researching object detection architectures	20
Setting up object detection system	5
Testing and validating navigation system	2

Fig 7.6: Sprint Table 3

Daniyal's tasks

Task	Duration(hours)
Literature review	10
Meeting with supervisor	8
Setting up initial code	12
Designing preliminary system architecture	10
Designing preliminary knowledge base for ethics engine	10
Installing and setting up TensorFlow	8
Developing a test model to test the GPU capabilities	2
Installing VSC, Python and setting up PyKnow	0.5
Designing a semantics graph for classifying objects the system should detect	0.5

Fig 7.7: Sprint Table 4

Task	Duration(hours)
Literature review	12
Meeting with supervisor	2
Redesigning system architecture	10
Designing and Implementing semantics graph	8
Data analysis	15
Coding and training of first iteration of navigation system	20
Validation of navigation system	4
Designing and implementing inter system communication	20
Researching appropriate cloud solutions for training	24
Debugging and diagnosing problems and issues with training runtime on cloud solution	72
Training the navigator system	72
implementing appropriate convolutional neural networks	10
Researching different convolutional neural networks	24

Fig 7.8: Sprint Table 5

Task	Duration(hours)
Literature review	12
Meeting with supervisor	8
Designing and implementing behavior modifiers for navigation system	20
Finalizing architecture	8
Data analysis	6
Coding and training of multiple iteration of navigation system	40
Validation of new iterations of navigation system	8
Coding ethics engine and navigation system communication	4

Fig 7.9: Sprint Table 6

Milestones

Sprint 1 milestone:

- Setting up the game and mods and testing their capabilities
- Setting up data collection environments and testing it

Sprint 2 milestone:

- Collect enough data points
- Preliminary data analysis
- 1st iteration of the navigation system and the ethics engine

Sprint 3 milestone:

- Finalized architecture
- Making sure all sub-systems communicate properly
- Having a working demo for CID

Gantt Charts

1 October – 30 October

Task number	Task Description	Start Date	Days to complete
1	Setting up initial code	1 October	5
2	Downloading and set up of simulation environments	2 October	4
3	Meeting supervisor	3 October	1
4	Setting up modding tools	6 October	3
5	1st iteration of ethics engine	7	4
6	Data points collection	9 October	14
7	Meeting supervisor	10 October	1
8	Testing viability of Google ML Platform	15 October	7
9	Designing preliminary system architecture	17 October	10
	Designing a semantics graph for classifying objects the system should detect	19 October	1
10	Meeting supervisor	24 October	1

Fig 7.10: Gantt table 1

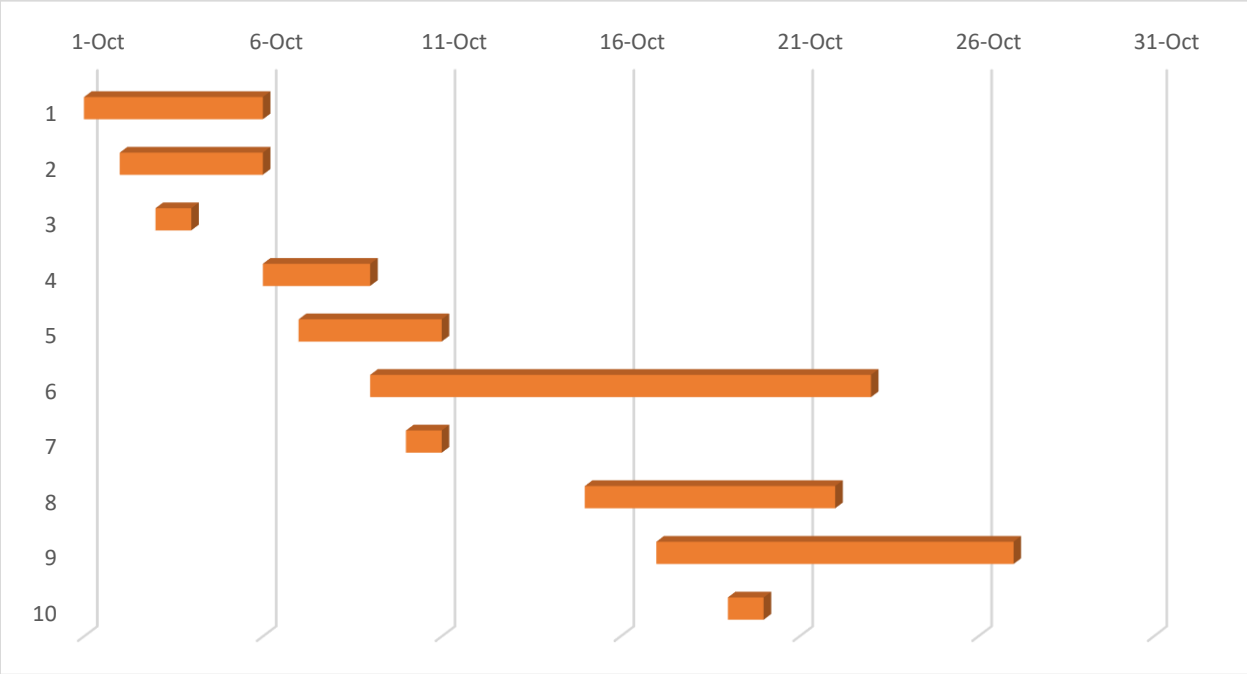

Fig 7.11: Gantt chart 1

1 November – 15 December

Task number	Task Description	Start Date	Days to complete
Task 1	Reinstalling GTA V and mods after malfunction	1 November	2
Task 2	Data points collection	3 November	12
Task 3	Processing data points	15 November	4
Task 4	Uploading data set to Google Cloud	19 November	5
Task 5	Redesigning system architecture	10 November	4
Task 6	Coding and training of first iteration of navigation system	16 November	6
Task 7	Training the navigation system	20 November	11
Task 8	Validation of navigation system	2 December	2
Task 9	Designing and implementing inter system communication	5 December	7
Task 10	implementing appropriate convolutional neural networks	6 December	4

Fig 7.12: Gantt table 2

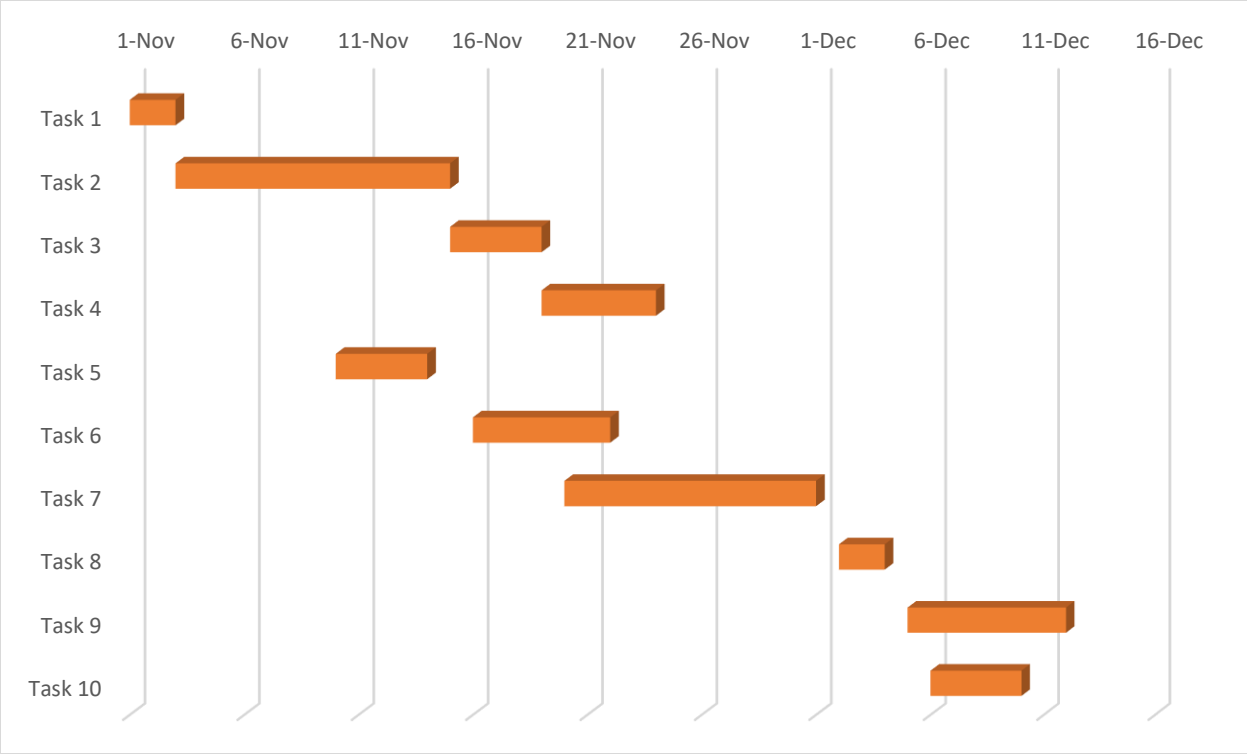

Fig 7.13: Gantt chart 2

15 December – 17 January

Task number	Task Description	Start Date	Days to complete
1	Data points collection	15 December	12
2	Researching object detection architectures	18 December	8
3	Setting up object detection system	26 December	2
4	Designing and implementing behavior modifiers for navigation system	29 December	10
5	Testing and validating navigation system	30 December	1
6	Finalizing architecture	1 January	4
7	Coding and training of multiple iteration of navigation system	25 December	16
8	Validation of new iterations of navigation system	10 January	4

9	Data analysis	14 January	3

Fig 7.14: Gantt table 3

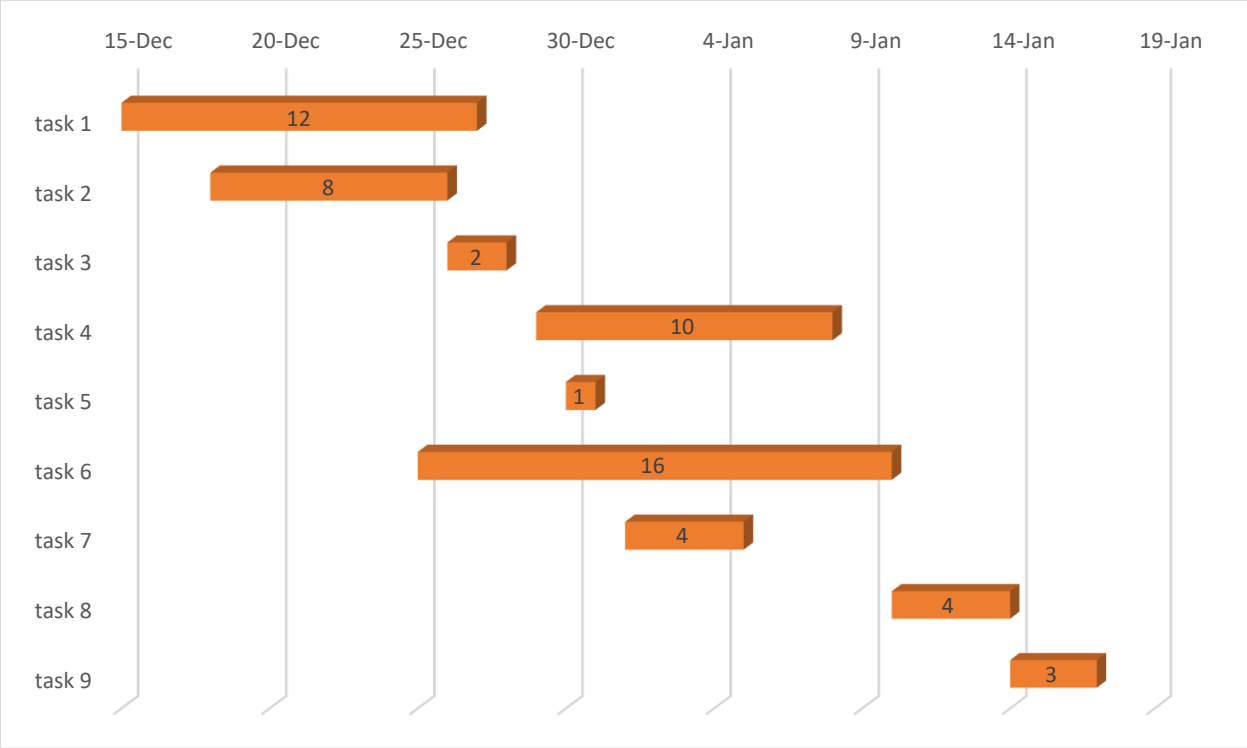

Fig 7.15: Gantt chart 3

8. Conclusion

Critical evaluation of the project and conclusions reached

This project is a large undertaking. It not only required a lot of research to be done, but also it needed the development of novel and innovative solutions for a lot of small problems that arose during its development.

Initially we wanted to go into detail of the implementation of each and every sub-system in CID. This was not possible. In reality we could only focus on the overall design and architecture of CID and how all the sub-systems within CID would communicate with each other.

An example of this would be the Semantics graph in the knowledge bas within the Ethics Engine sub-system. The semantics graph is a concept that we innovated specifically for the development of CID. It simplifies classification of objects and works well with the Object Detection system, and is an easy way of communication between the Ethics Engine and Object Detection systems. However, we could not develop this idea fully, as exploring this idea could be a full research project in it's own.

This is a somewhat reoccurring theme during the development of CID over the past semester. Each small solution and innovative idea (like the behavior modifiers), could be a full research project on it's own if we gave it's development more attention and manpower.

Thus, summarizing the project becomes very hard, because of the amount of work and ideas that have gone into it's development. However, we will revisit our objectives to help us draw meaningful conclusions.

1. A framework or approach that can make it easier to develop ethical autonomous driving systems

This objective was at the core of the development of CID. We took an outside in approach to the development of the sub-systems in CID. The ethics engine was treated as the heart of CID, and all the other sub-systems and their inputs and outputs were molded and modified so that a developer could implement his or her own ethical solution without having to modify CID.

Also, we majorly simplified the data collection and training process for CID, by limiting the scope of CID's ability to certain situations (CID can only drive in sunny weather, at noon in the middle of the city). This allowed us to iteratively increase the complexity that CID was exposed to, and made it very easy developed CID as we only needed to focus on one scenario at a time.

2. A framework or approach that can make it easier to modify and enhance pre-existing ethical autonomous driving solutions already built on the framework

This objective was reached by having a modular approach to the design of CID. The modular approach was something that we followed from the very start of development. This enabled us to use pre-existing solutions in our sub-systems (such as Object Detection), and to do rapid iterations of the sub-systems and the overall architecture of CID. When we retooled the architecture of CID during sprint 2, we had already started developing the navigator and ethics engine sub-systems. We did not need to start from scratch with these sub-systems, even when the whole architecture of CID was changed and simplified.

3. A framework or approach that makes it easier to standardize and to test ethical autonomous driving solutions

We achieved this objective by proving that a GTA 5 (a video game), is a suitable test bed and validation environment for an autonomous car. Using mods we can control the environment that CID is in at a very fine level. This makes it easier for us to test CID in different scenarios.

We also showed that an object detection system which is trained to work on images from real life, works perfectly fine when given images from the game. This proves that the game, at some level, is analogous to real life. Hence, it is a suitable testing environment for a system which might eventually be implemented on real roads.

At the beginning of the report, we stated that the development of CID would be like a case study. If 2 students (who are not exactly academic high achievers), with limited resources and expertise, can develop an autonomous car solution following our approach, then we can consider our approach as successful. Even though, due to hardware limitations, we cannot test CID fully in GTA 5, I am confident in stating that, within the duration of this semester, we were able to develop a somewhat comprehensive autonomous driving solution.

Retrospective of 1st semester

There were a lot of obstacles that were out of our control, that hindered the development of CID. For the first month we did not have access to the simulation environment we needed to begin development. Once we got access to the software we desired, we found out that the PCs available to us were not powerful enough to train the sub-systems of CID. This admittedly could have been discovered earlier by us, our mistake was to only do literature review while we waited for the software that we needed. It took us further 2 weeks to find a suitable free cloud solution for training CID.

We then discovered that we had miscalculated the amount of data we needed to collect in order to train the first iteration of CID (CID V0.01). This iteration failed miserably, leading to a complete change in architecture for CID.

The cloud solution also proved to be not reliable when training large amounts of data and training the data on complex models. This led to a lot of loss of data and progress. We also faced a setback when we discovered that a lot of our data had been deleted from the PC that we were using for development in RZ-11. This occurred mid December and caused further loss of progress.

Finally, we found out that the hardware resources available to us were not adequate to even full test CID.

In hindsight, we could have been more studious and researched the hardware requirements for the development of our project with more diligence. We simply cannot move forward without getting any hardware support from the university. With the current resources, the only possible progress for CID is minor improvements and bug fixes.

Future work

In the following semester, we would like to get better access to hardware resources to continue with the development of CID. We would also like to focus on the Behavioral Modifiers, and see how complex of a behavior we can get the navigator to execute based on the modifiers added to the input frame.

Furthermore, we would like to add predictive capability to CID. I.E, we want CID to have a sense of linearity and time. We plan on achieving this through the implementation of a HTM (Hierarchical temporal memory) model. HTMs take in stream of data that are being input to the system, and take in the state of the system itself, and then perform anomaly detection on that input. CID already defines the state of the input it sees (Object detection system), and it's own state (output from ethics engine and navigator systems). If we can buffer this data and send it as a stream to an HTM, we can then perform anomaly detection on the input of CID. We can then do a correlation study on the anomalies detected, and the current input of CID. If a certain input (such as a car merging into the lane front of CID) corresponds to an anomaly being detected, then we can do some predictive behavior based on this anomaly detection.

References:

1. A 'Brief' History of Neural Nets and Deep Learning, Part 2. (n.d.). Retrieved from http://www.andreykurenkov.com/writing/ai/a-brief-history-of-neural-nets-and-deep-learning-part-2/
2. Waymo – Waymo. (n.d.). Retrieved from https://waymo.com/
3. https://www.youtube.com/watch?v=LSX3qdy0dFg
4. How Googles Self-Driving Car Works - IEEE Spectrum. (n.d.). Retrieved from https://velodynelidar.com/lidar/hdlpressroom/How Googles Self-Driving Car Works - IEEE Spectrum.pdf
5. The everyday ethical challenges of self-driving cars | WTOP. (n.d.). Retrieved from https://wtop.com/national/2018/03/the-everyday-ethical-challenges-of-self-driving-cars/
6. Verification and Validation of Ethical Decision-Making in Autonomous Systems. (2017). *Modeling and Simulation of Complexity in Intelligent, Adaptive and Autonomous Systems 2017 (MSCIAAS 2017)*. doi:10.22360/springsim.2017.msciaas.001
7. Yilmaz, L., Franco-Watkins, A., & Kroecker, T. S. (2017). Computational models of ethical decision-making: A coherence-driven reflective equilibrium model. *Cognitive Systems Research,46*, 61-74. doi:10.1016/j.cogsys.2017.02.005
8. Reed, G. S., Petty, M. D., Jones, N. J., Morris, A. W., Ballenger, J. P., & Delugach, H. S. (2015). A principles-based model of ethical considerations in military decision making. *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology,13*(2), 195-211. doi:10.1177/1548512915581213
9. Holstein, T., Crnkovic, G. D., & Pelliccione, P. (2018). Ethical and Social Aspects of Self-Driving Cars. *EasyChair Preprints*. doi:10.29007/mgcs
10. Macal, C., & North, M. (n.d.). Tutorial on agent-based modeling and simulation. *Agent-based Modeling and Simulation*. doi:10.1057/9781137453648.0004
11. Large Scale Visual Recognition Challenge (ILSVRC). (n.d.). Retrieved from http://image-net.org/challenges/LSVRC/
12. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM,60*(6), 84-90. doi:10.1145/3065386
13. Overfitting and Underfitting With Machine Learning Algorithms. (2017, January 27). Retrieved from https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/
14. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. doi:10.1109/cvpr.2016.308
15. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM,60*(6), 84-90. doi:10.1145/3065386
16. Kumarsingh, B., Verma, K., & Thoke, A. S. (2015). Investigations on Impact of Feature Normalization Techniques on Classifier's Performance in Breast Tumor Classification. *International Journal of Computer Applications,116*(19), 11-15. doi:10.5120/20443-2793
17. Normalizations in Neural Networks. (2016, October 31). Retrieved from http://yeephycho.github.io/2016/08/03/Normalizations-in-neural-networks/

18. Tensorflow. (2019, January 18). Tensorflow/models. Retrieved from https://github.com/tensorflow/models
19. COCO - Common Objects in Context. (n.d.). Retrieved from http://cocodataset.org/#home

Additional Resources

- Hadrienj. (2018, August 27). Preprocessing for deep learning: From covariance matrix to image whitening. Retrieved from https://hadrienj.github.io/posts/Preprocessing-for-deep-learning/
- A Gentle Introduction to Transfer Learning for Deep Learning. (2018, December 12). Retrieved from https://machinelearningmastery.com/transfer-learning-for-deep-learning/
- Google Colaboratory. (n.d.). Retrieved from https://colab.research.google.com/notebooks/welcome.ipynb
- Stason, Stasonstason 8331819, Nguyễn Tài LongNguyễn Tài Long 3391210, Ajaychhimpa1Ajaychhimpa1 772, Shirish RanadeShirish Ranade 9917, Ivan_bilanivan_bilan 82311332, . . . SiddarthanSiddarthan 1. (n.d.). Google Colaboratory: Misleading information about its GPU (only 5% RAM available to some users). Retrieved from https://stackoverflow.com/questions/48750199/google-colaboratory-misleading-information-about-its-gpu-only-5-ram-available
- Das, S., & Das, S. (2017, November 16). CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more .... Retrieved from https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5
- Liang, J., & Liu, R. (2015). Stacked denoising autoencoder and dropout together to prevent overfitting in deep neural network. *2015 8th International Congress on Image and Signal Processing (CISP)*. doi:10.1109/cisp.2015.7407967
- http://cvml.ist.ac.at/courses/DLWT_W1
- (n.d.). Retrieved from http://www.softwaremetrics.com/fpafund.htm