Multi-Model For Driver Distraction Detection and Elimination

A Deep Learning Approach

github.com/DriverDistraction

driver.distraction.dl.app@gmail.com

https://www.youtube.com/channel/DDS

Driver Distraction System

1 Executive Summary (section 2)

This project aims to reduce number of car crashes by using AI systems that help the driver not to get distracted.

Our solution is to make everything inside the cabin accessible by voice (distraction elimination). Also we believe that our project would make it possible to ascertain quickly drivers' actions and their sights and to continually build use cases so we can understand when something may interfere with the driving process and create new ways to alert the driver (distraction detection).

Our project is composed of 5 AI/ML components:

1- Driver actions classification: which is done by a camera takes images of the driver and classify his action whether he's distracted or not.
2- Head pose estimation: which is done by another camera takes images of the driver's head and provide its angles.
3- Speech to text Engine: to generate text transcriptions from the driver voice to prepare it for the text classification model.
4- Text classification: which takes the text transcription and classify it to which command the driver wants to do.
5- Trigger word detection: to run the whole system by voice without getting distracted.

After a complete compression study of how we compress all these models together on a chip board we combined all these tasks together and we came up with a complete new multi-model system for driver distraction elimination and detection.

What's interesting about this project is that while developing, testing and enhancing we weren't only concentrating on the technicalities but we kept in mind the end goal of the project which is to create a business product that works offline to provide safety to the driver.

Thanks to Valeo Group Support Program we created such a product as they are one of the biggest companies regarding automotive suppliers that knows the target audience.

Table of Content

Table of Figures

2 Introduction (section 4)

2.1 Problem Statement

Figure 2-1 Driver Distraction Types

The number of road traffic deaths continues to increase, reaching **1.35 million** in 2016[1]. A research[2] conducted around traffic safety indicates that about **25% of car crashes have been caused by driver distraction**. Every year around 12,000 Egyptians lose their lives as a result of road traffic crashes, more than 20% of road accidents occur due to driver inattention/distraction.[3]

__Driver Distraction__ is a form of driver inattention that involves the diversion of attention away from safety critical activities within the driving task[4]. Driver distraction types are (Visual – Cognitive – Manual).[5]

2.2 Driver Distraction in Numbers (Statistics)

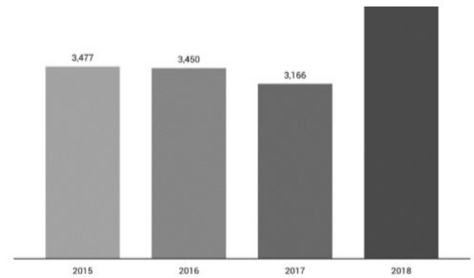


Figure 2-2 Distracted Driving Deaths in years

For many, driving is a daily activity, not requiring much thought or consideration. However, the sad reality is that there are 3,287 deaths each day due to fatal car crashes. On average, 9 of these daily fatalities are related to distracted driving

2.3 Current Solutions (Background Review)

Many solutions were developed to solve this rising problem using embedded systems and sensors but they were not as accurate and not very cost efficient.

2.3.1 Forward-Collision Warning (FCW)

It provides a visual, audible, and/or tactile alert to warn drivers of an impending collision with a car. Using sensors in front of the car to only alert if the car reaches a specific distance before collision

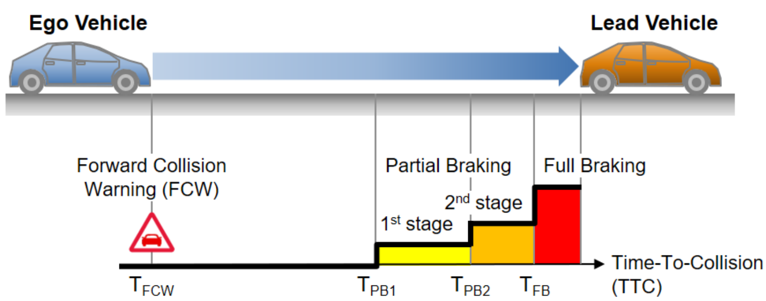


Figure 2-3 FCW and AEB Solution [ref]

2.3.2 Automatic Emergency Braking (AEB)

If the system senses a potential collision and the driver doesn't react in time, it engages the brakes. It uses the same tech as FCW.

Figure 2-4 Drivesafe.ly

1 World Health Organization
2 Observed by Lipovac et al.,
3 The Egyptian Central Agency for Public Mobilization and Statistics (CAPMAS)
4 (Regan et al., 2011; Young et al., 2008)
5 Automotive Resources International (ARI)

2.3.3 Drivesafe.ly and DriveMode Apps

This app is designed to read incoming text messages and email aloud for drivers so they keep their eyes on the road.

2.3.4 Apple IOS 11 and AT&T DriveMode

This latest operating system includes a Do Not Disturb While Driving mode (DND) that can block notification of incoming calls and texts when your iPhone senses driving motion or is connected to a car via Bluetooth. (It doesn't block functions that work through Apple's CarPlay system, such as music and navigation.) The DND feature can automatically send a text reply that says you're driving and will reply later. Phone calls are allowed if the iPhone is connected via Bluetooth.

Figure 2-5 AT&T DriverMode

2.4 Our Solution

Our Project will introduce deep learning into the equation to detect the driver actions and eliminate driver's distraction as a packed solution.

Detection: By analyzing driver actions and his head position the system will detect if the driver is distracted.
Elimination: By accessing every car functions (air conditioner, radio, etc.) by voice commands.

The system is a Standalone Offline End Device which doesn't need mechanical sensors as the system's backbone is Software-Oriented, not like in Forward-Collision Warning (FCW) and Automatic Emergency Braking (AEB). The compact system has vision and language processing combined as a packed solution which wasn't introduced before in any other software solutions Drivesafe.ly, IOS 11 and AT&T DriveMode apps.

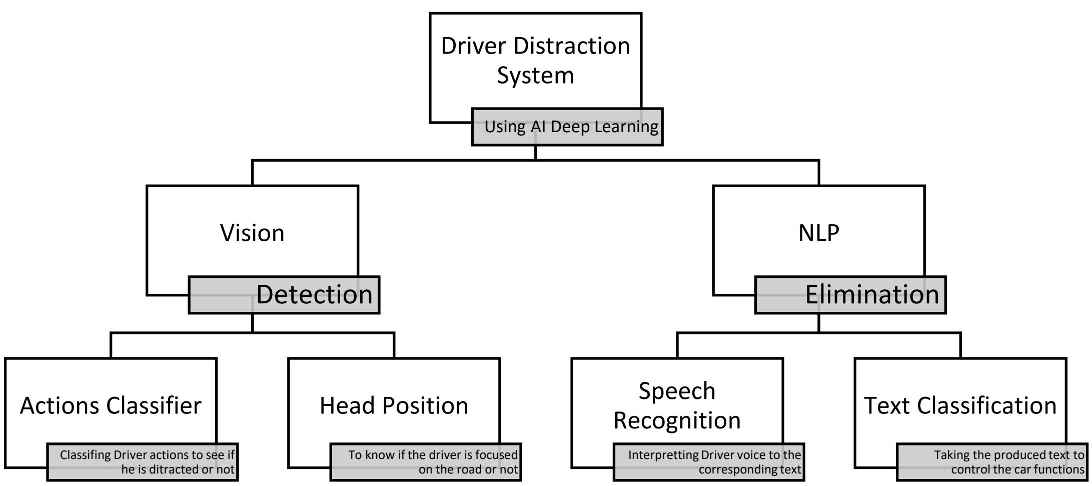


Figure 2-6 Our Solution Diagram

3 System Description (section 5)

3.1 High Level Explanation

3.1.1 Distraction Detection Subsystem

Composed of two AI/ML models (Head pose detection, Driver actions classification).

We need to monitor the driver and see if he is distracted or not (using a camera) by classifying the camera result to know if he is distracted or not, and as an added bonus we will classify what he is actually doing using classes of actions (Safe Driving – Texting – Talking on the phone – Operating the radio – Drinking – Reaching Behind – Talking to a passenger, ETC). Using a Deep Learning Neural Network. We also need to know the driver's eyes are on the road or not. This will be obtained by using a Head Posture Detection model

Figure 3-2 Driver actions classification

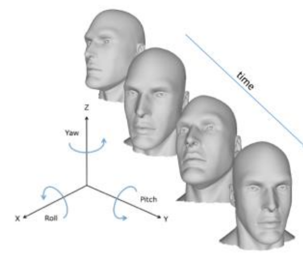


Figure 3-1 Head angles

3.1.2 Distraction Elimination Subsystem

Composed of two AI/ML models (Speech recognition, Text classification).

Introducing a Car Voice Commander to help take commands from the driver and take actions upon these commands without the driver using hands and losing the grip of the steering wheel.

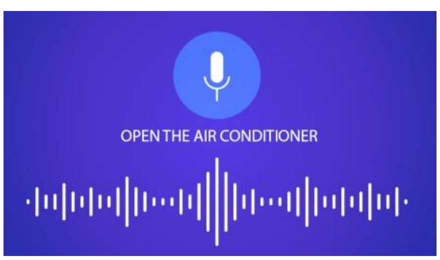


Figure 3-4 Speech recognition

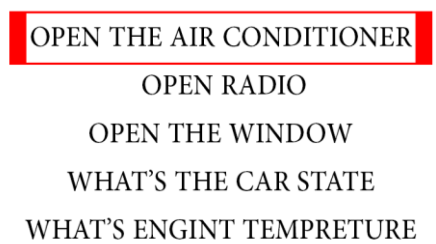


Figure 3-3 Command classification

3.2 System Requirements

3.2.1 Hardware Requirements

- 2 cameras and a mic and a flat screen to show results.
- GPUs to train our models.
- Nivida Jetson Nano board to run our model combined.

3.2.2 Software Requirements

- Our models trained and compressed.
- Pre-installed packages (Keras, Tensorflow, Pytorch, etc.)

3.3 System Constraints

The system must be installed on modern cars who has electronic control unit (Computer Box) and also the system requires two cameras, microphone and a flat screen.

	Driver Actions	Head Pose	Speech Recognition	Text Classification	Trigger word detection
GPU RAM usage	860 MB	1.4 GB	-	1.5 MB	153 MB

Table 1 System Configuration (RAM Usage)

The model runs on CPU real time but to get even faster response use GPU with minimum memory ram.

3.3.1 Technical Feasibility

User Familiarity: Low – Medium risk.

- The staff (Installing Engineers – Development Engineers) will require an easy training to interact with the system and technologies introduced.
- The driver will require an easy guide to use the system.

Hardware: Medium risk.

- GPU 4GB RAM Nvidia chip board.
- Two cameras, a mic and a flat screen.

Software: Low – Medium risk.

- The staff (Development Engineers) will require a strong background in the computer vision and NLP fields.
- The driver will smoothly use the system without any background knowledge.

Analyst and Management: Medium risk.

- Managing the developing team.
- Contact the suppliers and car manufacturers.

Cost: Low - Medium risk.

- Jetson Nano developer kit: 100$
- 2 Cameras: 80$
- Microphone: 35$
- Connectors: 30$
- The System will totally cost 250$

3.4 More Detailed Explanation

From the end user (driver) perspective, there will be two cameras pointed at the driver one to detect his action and the other to detect his head position. Also the voice assistant services will be available if the trigger word detected which is 'Activate'. After taking the driver's commands the system will start to analysis and classify the commands and respond with the proper action.

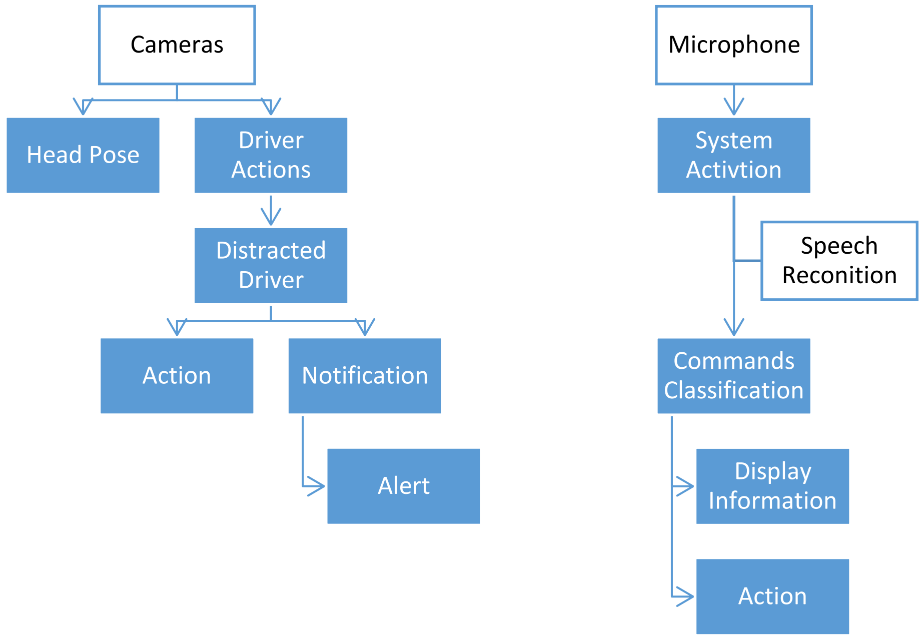


Figure 3-5 System Flow

4 System Architecture (section 6)

4.1 Use Case diagram

To identify the user interaction with the system we used the use case diagram as follows:

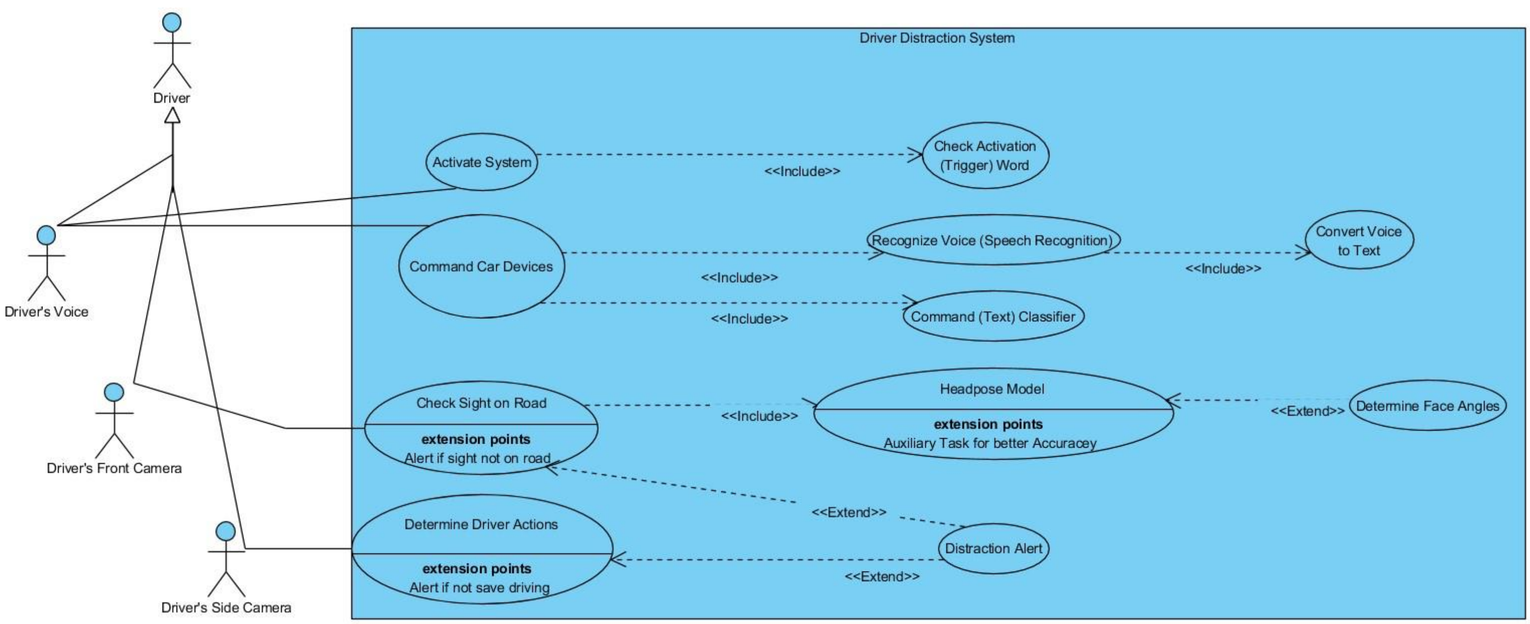


Figure 4-1 System Use Case Diagram

4.2 Sequence Diagram

To identify how the system components are arranged in time sequence we used the sequence diagrams as follows:

4.2.1 NLP Models

Shows how NLP models are used from system activation through speech recognition till they give the final actions needed.

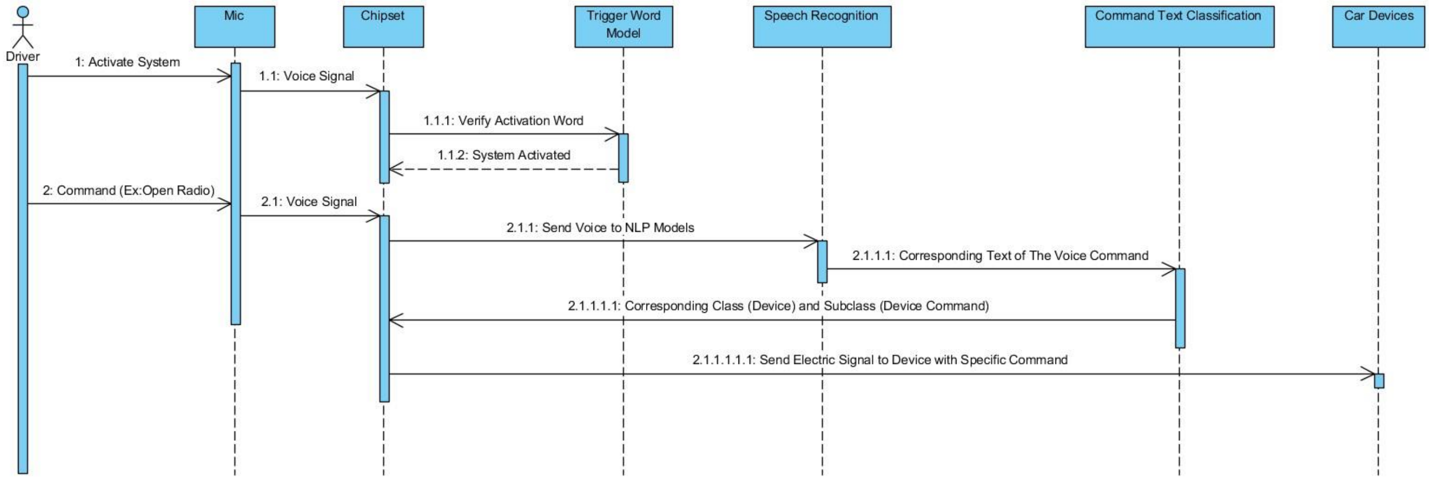


Figure 4-2 NLP Models Sequence Diagram

4.2.2 Vision Models

Shows how to utilize cameras to determine whether the driver is distracted or not.

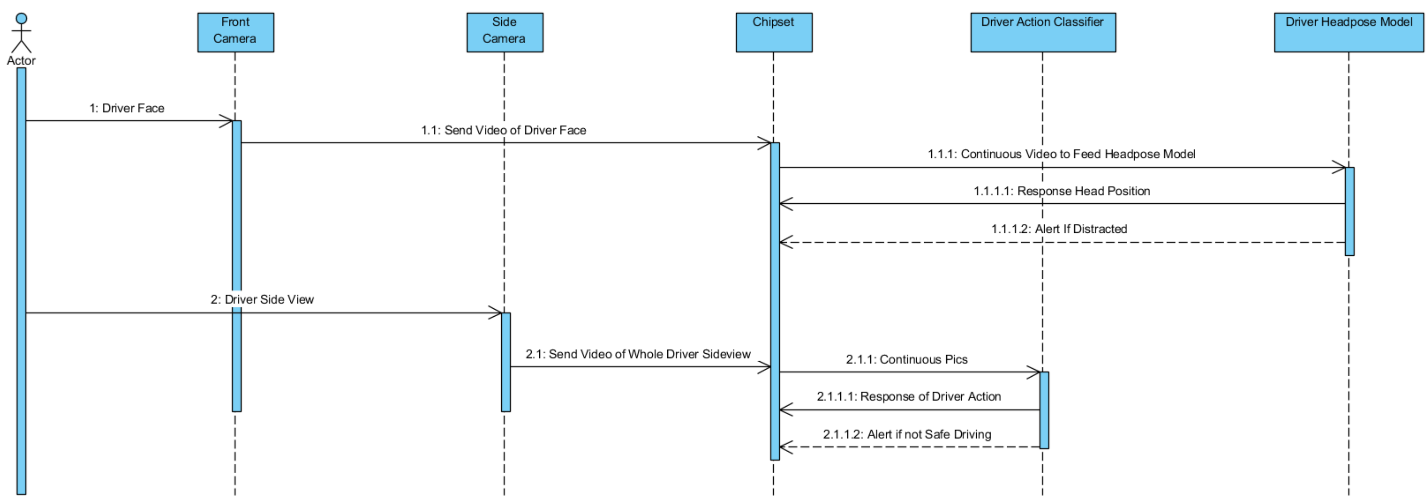


Figure 4-3 Vision Models Sequence Diagrams

4.3 Software Architecture Diagram

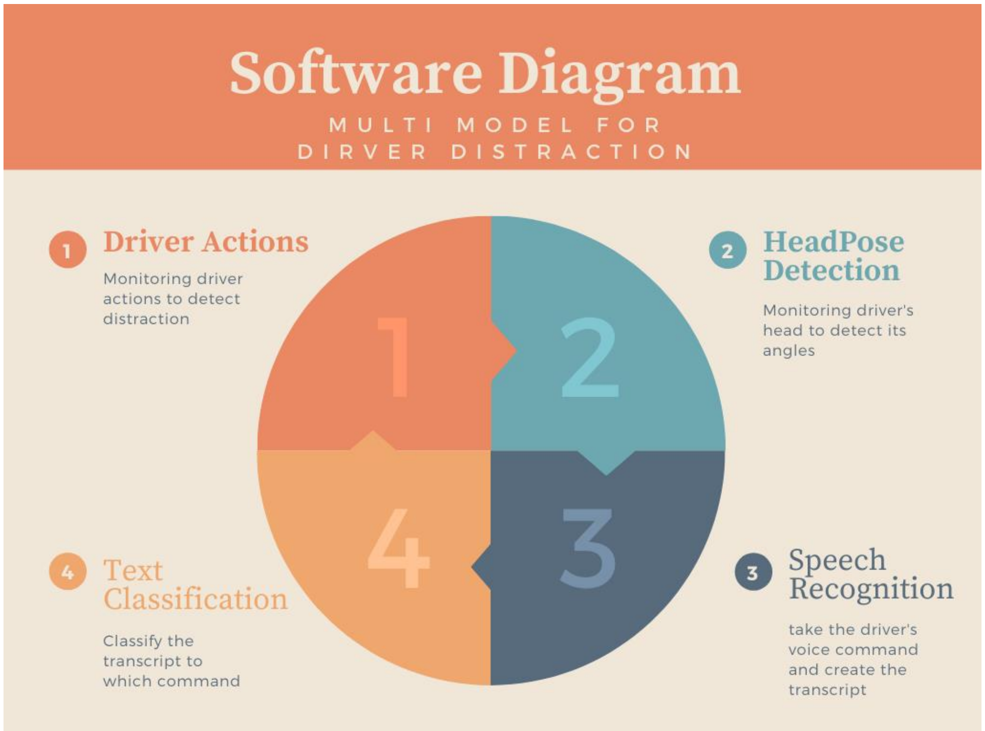


Figure 4-4 Software Components

4.4 Hardware Architecture Diagram

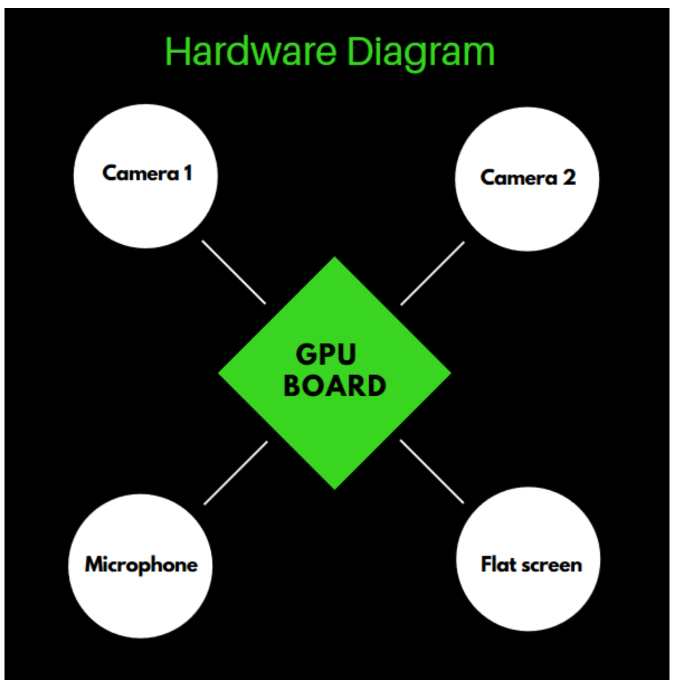


Figure 4-5 Hardware Diagram

5 System Implementation (section 7-a)

As mentioned before the system composed of 5 AI/ML main components. Let's discuss them with some details.

note in this section we will discuss only concepts and the implementation, models evaluation and testing will be discussed in the next section.

5.1 Driver Actions Classifications

There is a camera positioned on the right side of driver. It takes a video and send it to a chip i.e. Jetson Nano and the chip process the video frame by a frame and tell what the state of the driver is between 10 different classes. And if the driver is distracted the chip sends signal to the car's embedded systems to take a proper action or just alert the driver.

5.1.1 Dataset

State Farm dataset[6] contains snapshots from a video captured by a camera mounted in the car. The training set has 22.4K labeled samples with almost equal distribution among the classes for the 25 different driver in training set. In the Test set there is 79.7K unlabeled test samples. There are 10 classes of images (Safe driving, Texting L&R, Drinking, etc.) so we have 9 different classes of distraction and only one not distracted class which is safe driving.

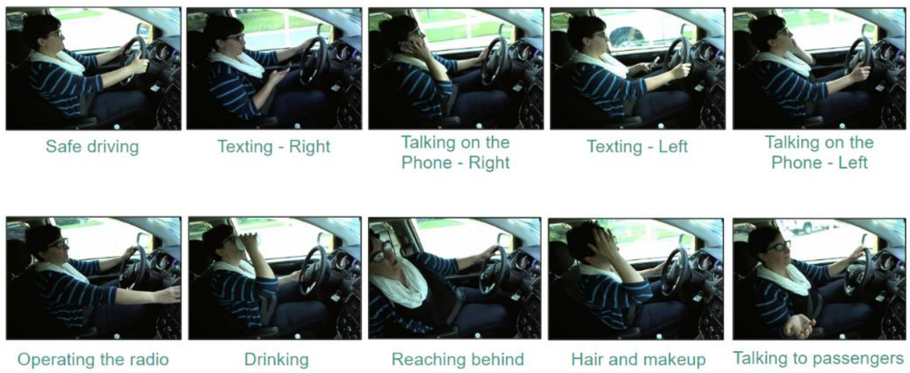


Figure 5-1 Driver Actions Classes

AUC Distracted Driver Dataset[7] contains 17.3K snapshots also from a camera. It has the same classes as the StateFarm dataset with 31 different drivers.

5.1.2 Model Architecture

With the understanding of what needs to be achieved, we proceeded to build the CNN models from scratch. We added the usual suspects — convolution batch normalization, max pooling, and dense layers. Surprisingly we get validation score 99.6%! We checked if there is leakage and Of course there was. So We split the data by driver id and if training the data on the same model we get 30% only. So, we used transfer learning using ResNet50 and froze 50% of layers. We tuned the model adding Regularization, Dropout, Small learning rate and Augmentation. Beside adding 13.7K photos from the AUC dataset.

Figure 5-3 Auto generated different augmentations on the same photo

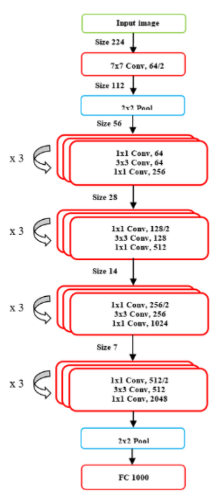


Figure 5-2 ResNet50 Architecture

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 62, 62, 32)	896
batch_normalization_1 (Batch	(None, 62, 62, 32)	128
conv2d_2 (Conv2D)	(None, 62, 62, 32)	9248
batch_normalization_2 (Batch	(None, 62, 62, 32)	128
max_pooling2d_1 (MaxPooling2	(None, 31, 31, 32)	0
conv2d_3 (Conv2D)	(None, 31, 31, 64)	18496
batch_normalization_3 (Batch	(None, 31, 31, 64)	256
conv2d_4 (Conv2D)	(None, 31, 31, 64)	36928
batch_normalization_4 (Batch	(None, 31, 31, 64)	256
max_pooling2d_2 (MaxPooling2	(None, 16, 16, 64)	0
conv2d_5 (Conv2D)	(None, 16, 16, 128)	73856
batch_normalization_5 (Batch	(None, 16, 16, 128)	512
conv2d_6 (Conv2D)	(None, 16, 16, 128)	147584
batch_normalization_6 (Batch	(None, 16, 16, 128)	512
max_pooling2d_3 (MaxPooling2	(None, 8, 8, 128)	0
flatten_1 (Flatten)	(None, 8192)	0
dense_1 (Dense)	(None, 512)	4194816
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 10)	5130

Total params: 4,488,746
Trainable params: 4,487,850
Non-trainable params: 896

Figure 5-4 Driver Actions Classifier Architecture

6 State-Farm dataset [link]
7 AUC Distracted Driver Dataset [link]

5.1.3 Model Results

Figure 5-5 Driver Actions Detected

5.2 Head Pose Estimation

Traditionally head pose is computed by estimating some key-points from the target face and solving the 2D to 3D correspondence problem with a mean human head model[8]. We argue that this is a fragile method because it relies entirely on landmark detection performance, the extraneous head model and an ad-hoc fitting step. We present an elegant and robust way to determine pose by training a multi-loss convolutional neural network on 300W-LP, to predict intrinsic Euler angles (yaw, pitch and roll) directly from image intensities through joint binned pose classification.

5.2.1 Dataset

For the training dataset, 300W-LP dataset[9], a synthetic expansion of the 300W dataset. Augmentation of 300W was performed in order to obtain face appearances in larger poses. This dataset provides annotations for both 2D landmarks and the 3D projections.

Figure 5-6 300W-LP Dataset

8 Fine-Grained Head Pose Estimation Without Key-points Paper [link]
9 300W-LP dataset [link]

Test set information, AFLW2000-3D dataset[10], consists of 2000 images that have been annotated with image-level 68-point 3D facial landmarks. This dataset is typically used for evaluation of 3D facial landmark detection models. The head poses are very diverse and often hard to be detected by a CNN-based face detector.

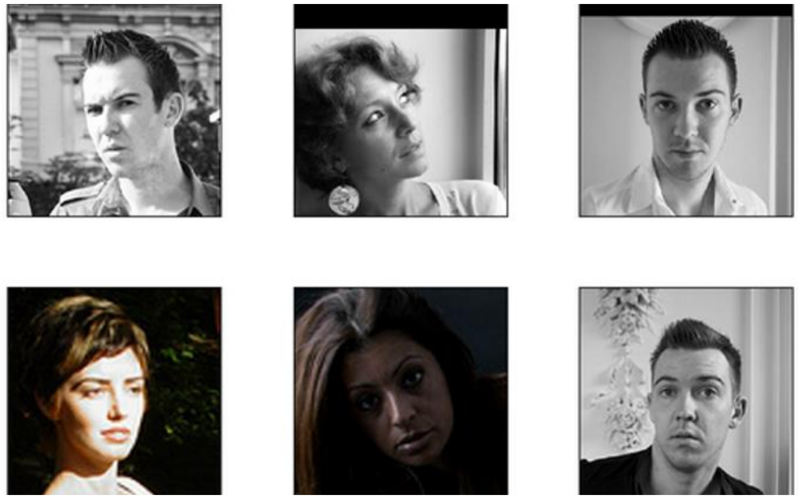


Figure 5-7 AFLW2000-3D Dataset

5.2.2 Model Architecture

First thing is face detection which is done before entre the very first layer of our model architecture. Then the face goes through ResNet50 (Fig. 5-8). Then the architecture divided to 3 branches every branch is responsible for an angle of the three angles. Each branch goes through a classification phase in which we have 66 classes! But why 66 classes? Because every class represent a 3 degrees of the 198 degrees that an angle can be set! The minimum degree that an angle can be set to is -99 and the maximum degree is +99 so we have 198 degrees of each angle! First it goes through a softmax which classify the angle to which 3 degrees it could be then goes through mean square error (MSE) as it detects which degree specifically the angle is. This happens in each of the three branches of the model.

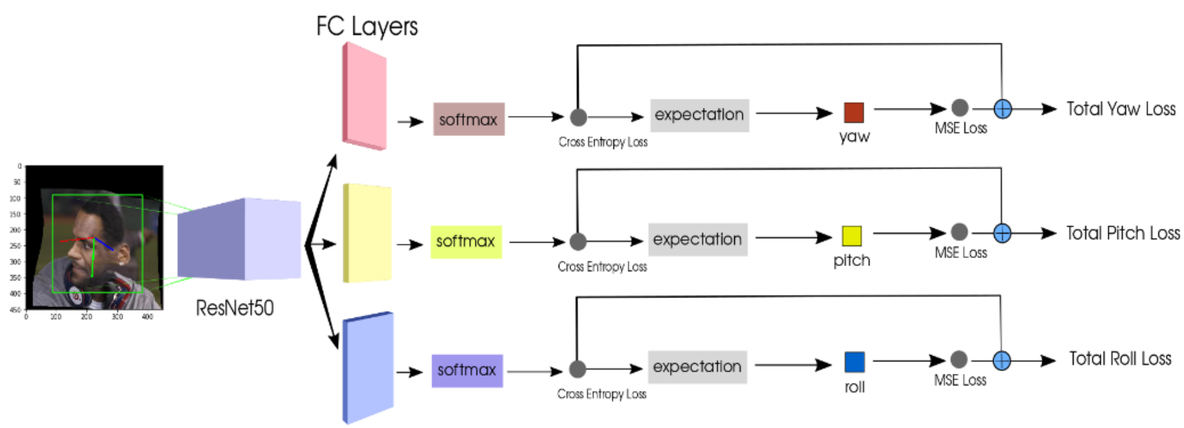


Figure 5-8 Head Pose Model Architecture

5.2.3 Model Results

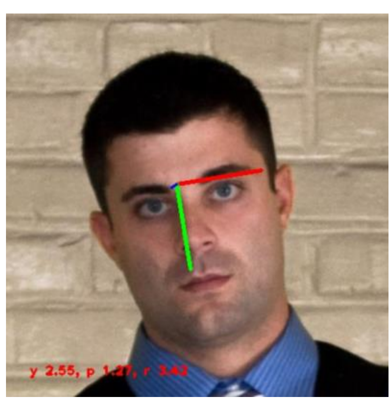


Figure 5-9 Results of Head Pose Model

10 AFLW2000-3D dataset [link]

5.3 Speech Recognition

Automatic Speech Recognition (ASR) is the process of deriving the transcription (word sequence) of an utterance, given the speech waveform. Speech understanding goes one step further, and gleans the meaning of the utterance in order to carry out the speaker's command.

We Used Deep Speech, a Deep Neural Network speech recognition system. We chose Deep Speech as it's an open source project that is active and maintainable and adaptable with the state of the art technologies in speech recognition to enhance the accuracy in every release, and also they provide an easy to use modular API, and an open source model that was trained in almost 100 thousand hours of voice for a month in a lot of GPU, as training speech recognition system require a lot of resource which we don't have, so it will be impossible to us to train our model form scratch and achieve a usable accuracy. And here's the overall pipeline:

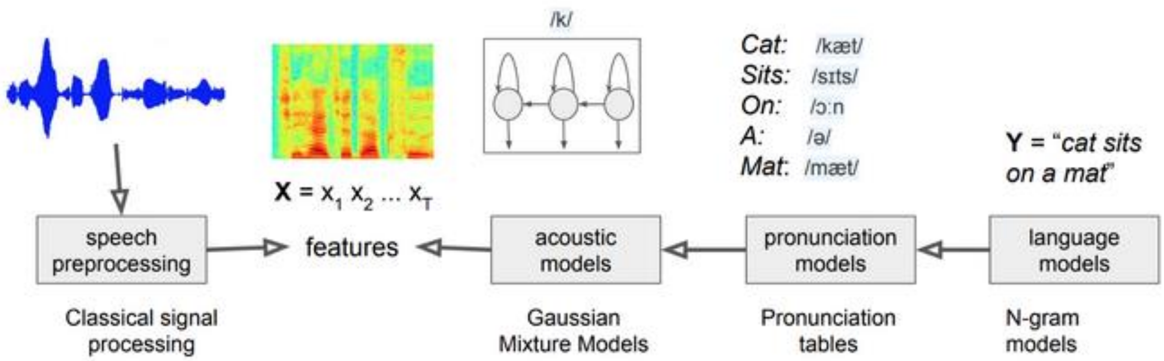


Figure 5-10 Deep Speech Pipeline

5.3.1 Model Architecture

At each time step, Deep Speech applies three FC layers to extract features. Then, it is feed into a bi-directional RNN to explore the context of the speech. Deep Speech adds the results from the forward and backward direction together for each time step. Then, it applies another FC layer to transform the result. Finally, the probability for each character at each time step is computed with a softmax.

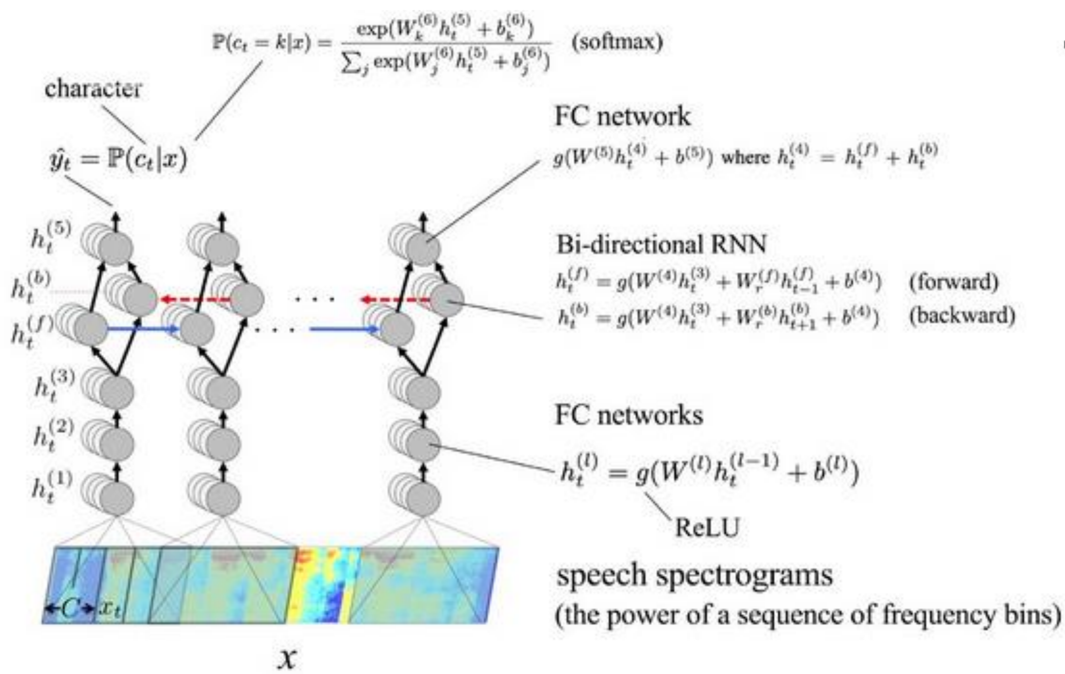


Figure 5-11 Deep Speech Model Architecture

Deep speech uses the objective function below to find the optimal sequence of characters. This objective function is based on the distribution output of the deep network, an N-gram language model and the word count of the decoded sentence. The language model allows us to produce grammatically sound sentences. But the deep network may be biased to create un-necessary more or fewer words than it should be. So we add a word count objective to tune the selection process.

The Deep speech model utilize a language model to generate a words close to the real used words, as the acoustic model predict the word character by character so the language model tries to output words that are close to real words.

hyperparameters

$$\underbrace{Q(c)}_{\text{a sequence of characters}} = \log(\mathbb{P}(c|x)) + \alpha \log(\underbrace{\mathbb{P}_{\text{lm}}(c)}_{\text{language model}}) + \beta \; \text{word\_count}(c)$$

language model

probability of c according to a N-gram model

Figure 5-12 Deep Speech Objective Function

And also the language model can increase the probability of outputting a specific words based on the words that the language model Is trained on, so we can utilize this feature by trying to limit the output words to the words that are more likely to be used in our car commands, so we can increase the accuracy of the prediction of the Deep speech model in our task.

That's why we created car commands dataset and fine-tuned the general language model of the deep speech model to our dataset to resize the output domain of words so the predicted sentence would be most likely a command to the car.

5.4 Text Classification

After the ASR convert the voice command to text, we will then classify the text to specify the desired command from a predefined set of commands and we will also extract the information from text that will be relevant to the command.

5.4.1 Dataset

The predefined commands are the most common commands used in the car, for example (turn on the radio, Set the air conditioning to 55 degrees, etc.). And we couldn't find a well-suited dataset for our classifier model so we had to create our own simple dataset.

The dataset consists of (Command – Subcommand – Sentence) as the command represent the object and the subcommand is the action desired from that object, i.e. (Set the air conditioning to 55 degrees, Command: Air Conditioner, Subcommand: Set)

Sentence	Command	Subcommand / Entity
turn on the radio	Radio	radio on
radio on	Radio	radio on
turn the radio on	Radio	radio on
play the radio	Radio	radio on
play radio	Radio	radio on
listen to the radio	Radio	radio on
resume radio	Radio	radio on
resume	Radio	radio on
turn off the radio	Radio	radio off
radio off	Radio	radio off
turn the radio off	Radio	radio off
pause	Radio	radio off
pause the radio	Radio	radio off
stop	Radio	radio off
stop the radio	Radio	radio off
radio next channel	Radio	next cahnnel
change the radio channel	Radio	next cahnnel
change the radio	Radio	next cahnnel
next radio channel	Radio	next cahnnel
next channel	Radio	next cahnnel
change channel	Radio	next cahnnel
channel up	Radio	next cahnnel
radio channel up	Radio	next cahnnel
radio previous channel	Radio	previous channel
radio last channel	Radio	previous channel

Figure 5-13 Car Commands Dataset

Using Sklearn as it's very powerful and fast in simple text classification problems with no need to use Neural Networks (RNN, LSTM, etc.) as it uses simple machine learning methods in classification (SVM, SGD, Naïve Bayes, etc.) with more in detail.

5.4.2 Support Vector Machine

SVM or Support Vector Machine is a linear model for classification and regression problems. The idea of SVM is simple: The algorithm creates a line or a hyperplane which separates the data into classes. If there are more than one class, the model uses n classes and n separators and uses One-Vs-All to classify a class from all other classes and then does that n times to create and train n classes.

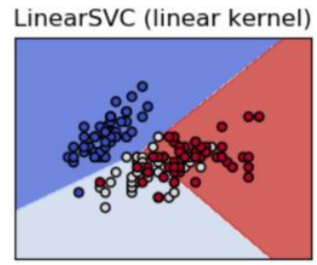


Figure 5-14 LinearSVC

After many experiments on the optimizers and data augmentation we started classifying the sentences but since the sentence may have some value we needed name entity recognition. Ex: "set the air conditioner to <u>16 degrees</u>".

5.4.3 Name Entity Recognition

NER is a subtask of information extraction that seeks to locate and classify named entities mentioned in unstructured text into pre-defined categories such as person names, organizations, locations, medical codes, time expressions, quantities, monetary values, percentages, etc.

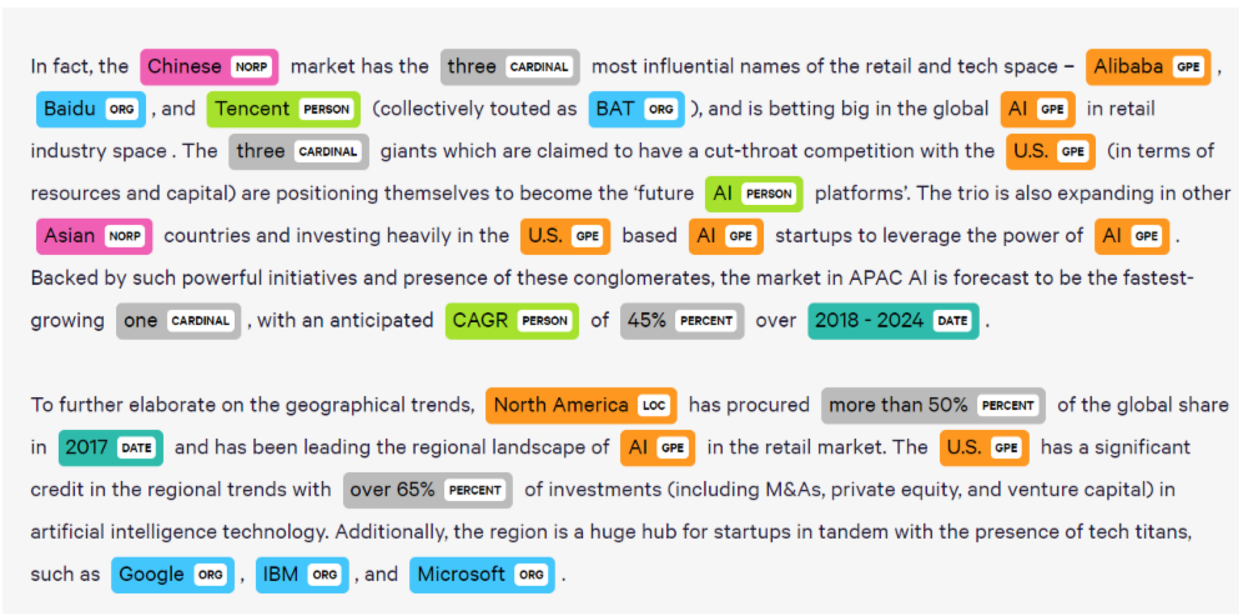


Figure 5-15 Name Entity Recognition

6 Hardware Implementation (section 7-b)

Due to the Covid19 situation, we had some challenges shipping the chip board as planed in our contingency plan that we presented in the last submission.

We started enhancing and testing the models to work on CPU real timing as a replacement of the Nvidia Jetson Nano chip board.

And here’s our Gantt chart:

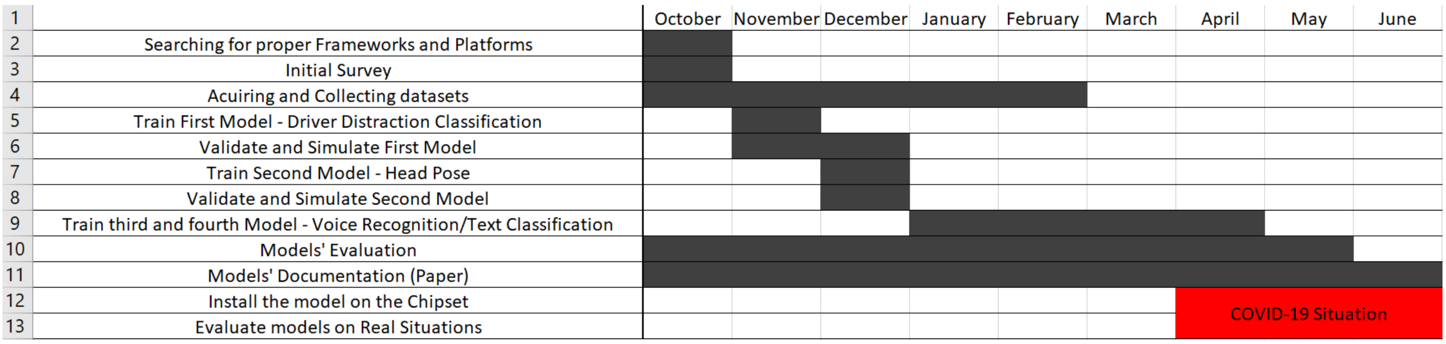


Figure 6-1 Gantt chart

And here’s our Risk and contingency plan:

Scenario	Problem / Future Risk	Recovery
NLP Dataset and Language Model	We couldn’t find a specific dataset for our problem of car commands	Creating our own dataset
NLP Models Training	The time for training is very huge almost a month and will require a lot of GPUs	We will use a pre-trained model that is provided by deep speech team
NVIDIA Jetson Board shipment	Not available in our country, can’t arrive on time	We will acquire it from Amazon, we will try to borrow one from other people
Chipset Installment	Mic, Cameras and models will need an integrated GPU	Mics and Cameras Interfaces to the NVIDIA Jetson Board
Camera and Mics	Our system needs clean audio and clean Picture Frames	We will acquire Cameras and Mics with a nice performance from Amazon

Figure 6-2 Contingency Plan

7 User interface (section 7-d)

That will be the wakeup/trigger word detection as that it’s the only part the end user will deal with. When the trigger word is said the system will wake up and start listening to the user commands and process them.

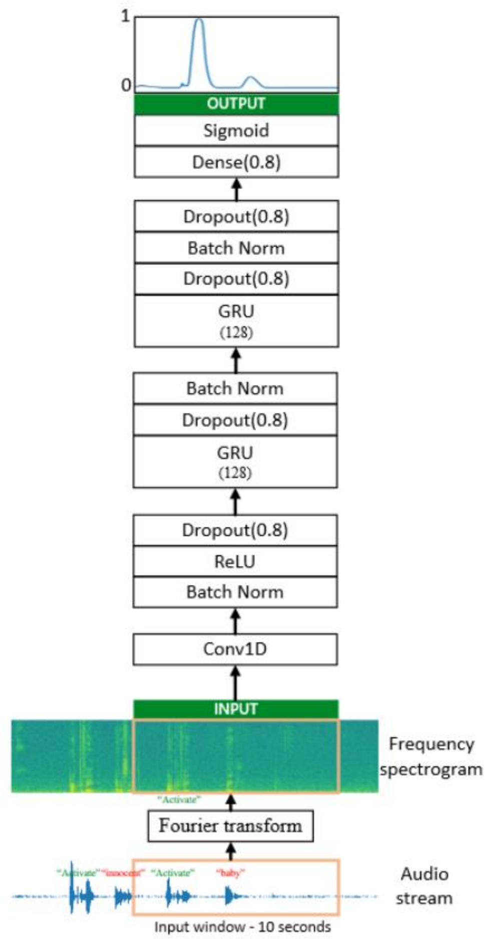


Figure 7-1 Trigger Word Detection Architecture

For the sake of simplicity, let's take the word "*Activate*" as our trigger word. The training dataset needs to be as similar to the real test environment as possible. For example, the model needs to be exposed to non-trigger words and background noise in the speech during training so it will not generate the trigger signal when we say other words or there is only background noise. We generated different kinds of audio recordings with different kinds of backgrounds, after generating the speech dataset we had to transform the audio recordings to spectrogram using Fourier transformation to construct a proper format to be the input of the model.

After that we generated our full training set and development set and pushed them to the model. We started the training on a proper GPU for a proper amount of time. After finishing the training, we started to test our model and make predictions. And at last when we had a good model evaluation we started the live testing with our own voice and environment. And when everything is set we linked the whole model with the other models in the project. So now when this model triggered by the ‘Activate’ word the whole system is working.

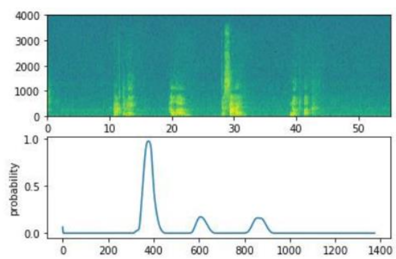


Figure 7-2 Wake up Word Detected

We also add the silence detection mechanism to skip prediction if the loudness is below a threshold, this can save some computing power.

8 Testing and Performance Evaluation (section 8)

8.1 Driver Actions Classification

We used F1 score and confusion matrix for model evaluation.

	Accuracy	Precision	Recall	F1
Driver Actions	95.4%	94%	94%	94%

GPU RAM usage 860 MB after Pruning compression GPU RAM usage 265 MB.

Confusion matrix:

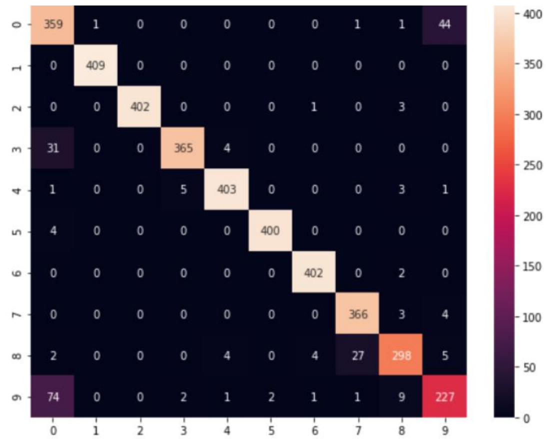


Figure 8-1 Driver's Classes of Actions Confusion Matrix

Enhancement idea: Autoencoder to make the unlabeled data useful as we got lots of unlabeled data.

8.2 Head Pose Detection

We used mean absolute error (MAE) for model evaluation.

	Yaw	Pitch	Roll	MAE
Face's Angles	7.31	6.31	5.04	6.21

CPU RAM usage 1.4 GB after a complete compression study the best way to compress while saving the mean absolute error from exploding was Quantization we got GPU RAM usage of 466 MB.

Enhancement idea: sometimes adding axillary task could help the main task learning, we believe if we added detecting the face could help detecting the angles.

8.3 Speech Recognition

We used word error rate (WER) metric to evaluate speech recognition performance. The model trained on American English which achieves a 7.5% word error rate on the LibriSpeech clean test corpus.

8.4 Car Commands Classification

	Accuracy	Precision	Recall	F1
Car Commands	84.35%	86%	82%	84%

8.5 Trigger Word Detection

Real Time testing Accuracy: 95%

Enhancement ideas: Adding more background sounds to the training data could help better defeat the noise.

8.6 Models Response Time

	Driver Actions	Head Pose	Speech Recognition	Text Classification	Trigger word detection
CPU	0.11 sec	0.37 sec	Real Time	0.005 Sec	Real Time
GPU (GTX 1050TI)	0.0618 sec	0.019 sec	Real Time	-	Real Time

9 System Usability and Social Impact (section 9)

9.1 Usability

To know if the system is usable from both a user and a developer perspective, we will use five main elements:

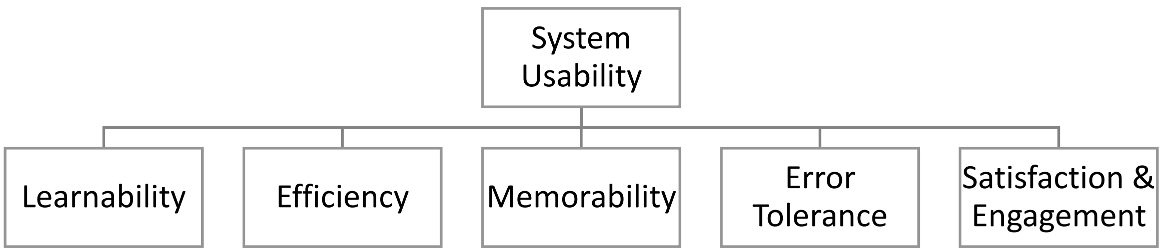


Figure 9-1 System Usability Elements

9.1.1 Learnability

Since is our system is automated, the user won't take much time to learn how use it as he/she will only activate the system using voice to use the command system (which is pretty popular and easy to use like Siri), and the vision part will work on background continuously without knowledge and will only alert him if he/she is distracted.

9.1.2 Efficiency

The system is an open source system and works offline with no internet. The system is easy to install to pretty much any car and will get sufficient results in any environment as:

- The vision datasets have variable situations (Day – Night) and camera positions and noise thanks to the augmentation methods we used. Also, the system was fed by multiple datasets to improve overall accuracy and reliability.
- The NLP dataset was created by us to fit the situation and covers each command with multiple sentences to assure variation. And the DeepSpeech API was trained to adapt for noise using its augmentation techniques.

9.1.3 Memorability

The user doesn't need to memorize much as the system is automated to eliminate the user functions in the car and to detect if he/she is distracted or not. So, it is a straight forward system.

9.1.4 Error Tolerance

To classify if the user in vision models is distracted or not, we used balanced datasets as previously mentioned. And tested our system using another dataset to assure that our model is fine tuned to any situation. Then we tested it in Real-Time environment to make sure it works fine and it gave great results as we reviewed with our mentor.

For NLP models, Activation System and DeepSpeech API are huge systems and we fine-tuned them to work with our specific model.

9.1.5 Satisfaction & Engagement

Due to COVID-19 Situation, we didn't spread the system to multiple users to get various feedbacks. But we still managed to install it to our cars (Using Computer CPU) to get an estimate of how it will work. And results were satisfying.

9.2 Product Life Cycle and Impact

The next section was planned before COVID-19 situation to market for the product and to see impact (socially and economically)

9.2.1 Initiation – Introduction

After searching how common driving accidents are. And how driver distraction is one if not the main reason for causing terrible accidents.

And after searching and experiencing most common driver distraction solutions, we came to conclusion that deep learning model as discussed would be a best fit nowadays, by deploying the aspects of prevention using voice commands and detection using image recognition and action classification.

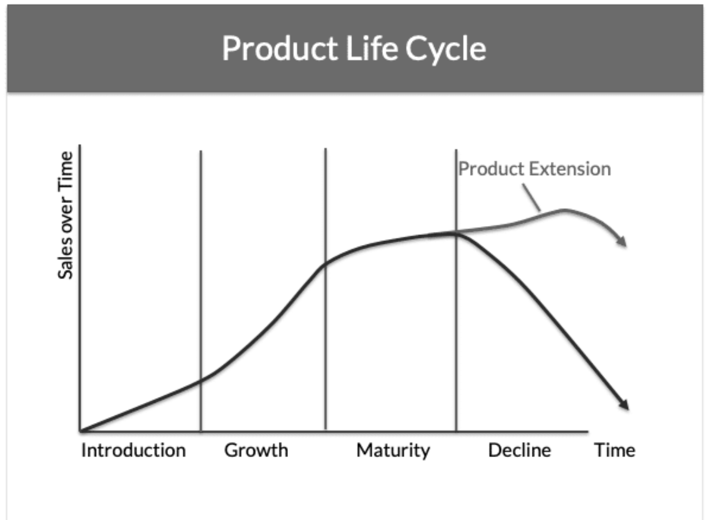


Figure 9-2 Product Life Cycle

9.2.2 Problem Child – Growth

It is the question of how our solution will fit the market share to grow.

Most solutions in the market are focused on only just one aspect just as discussed before. And they only properly focus on distraction detection. But our solution adds prevention as a bonus.

Since our solution is a standalone solution that can be easily integrated in any car as it doesn't require internet (just a camera – mic and the chip) and it will produce signals to control the car functions.

9.2.3 Maturity

We are trying to propose an MVP that is updated on the go (Producing minimum Features – testing – adding more features) cycles.

9.2.4 Decline

To extend our project cycle we will add more features to tackle cognitive distraction. And trying to find and acquire cognitive datasets. And this will impressively extend the life span of our solution.

However, with the revolution of self-driving cars that will take our project hopefully as an initial point to go from there, our project will be declined as a standalone solution and it will be converted to a self-driving cars' feature.

9.3 Impact

To measure impact of our project we used the following elements.

9.3.1 Socially

By reducing car accidents, the roads will be safer so roads won't need further maintenance. Also, drivers and passenger will feel safer and hopefully the death rate will decrease due to this affordable system.

9.3.2 Environmentally

The system won't rely on sensors (radiation) neither mechanical components (as in FCW and AEB).

9.3.3 Commercially and Economically

This system's main logic relies on software which won't cost much. And as the system Hardware, it only needs a 2-Camera setup, a Mic and a Jetson NVidia Chipset to hold the four models. Also it doesn't need internet connection, so it will be easily integrated with any car.

9.3.4 Industrial

Targeting Big car companies (Mercedes, BMW, …) to integrate the proposed system to new cars and car Technology suppliers.

9.4 Considerations

As our system is straight forward and simple. Ethical and Legal considerations are minimum.

9.4.1 Legal Considerations

The system has two cameras and a Mic that record driver's movement and voice, and a GPU board that maybe hard to ship in some countries.

9.4.2 Ethical Considerations

The continuous recording by the system cameras and mics may face some ethical problems. No health concerns are encountered in this project.

10 Conclusion and Future Work (section 10)

The system based on deep learning to make everything inside the cabin accessible by voice (distraction elimination). Also we believe that our project would make it possible to ascertain quickly drivers' actions and their sights and to continually build use cases so we can understand when something may interfere with the driving process and create new ways to alert the driver (distraction detection). After a complete compression study of how we compress all these models together on a chip board we combined all these tasks together and we came up with a complete new multi-model system for driver distraction elimination and detection. We kept in mind the end goal of the project which is to create a business product that works offline to provide safety to the driver.

Future Work:

- The system only includes a Voice Commander; we can add a Question answering model to make it Voice-assistant alike.
- Adding Emotion Detection System to eliminate the Cognitive Part of Driver Distraction.
- Some enhancement of the system structure is referred in section 8.

11 Appendix

11.1 Project Management Methodology (Appendix A)

We used multiple methodologies to fit the project requirements in specific times as follow

11.1.1 Waterfall for Big Picture

To make sure we don't fall behind. We planned the system headlines and models far ahead from the beginning (October 2019) also to make sure we have a proposed document for findings, competitions and mentorship programs. The Gantt chart plan is referred in (Appendix B).

11.1.2 Agile for developing the first model

The first model we developed was the Vision model (Drive Distraction System) back in November, and to make sure that each team member has a solid understanding of the system, we all worked collaboratively in this model. By using multiple architectures and training techniques.

This methodology helped us to have a better understanding of Deep Learning and Machine Learning techniques that we will use in the next models.

The first model was developed through cycles of training and testing. And once we had a good accuracy to move on to the next model, we deployed it. (Agile) for example we used ResNet Arch (18 – 50 - 151) and each team member's responsibility is to develop the model to have a better understanding of the whole picture.

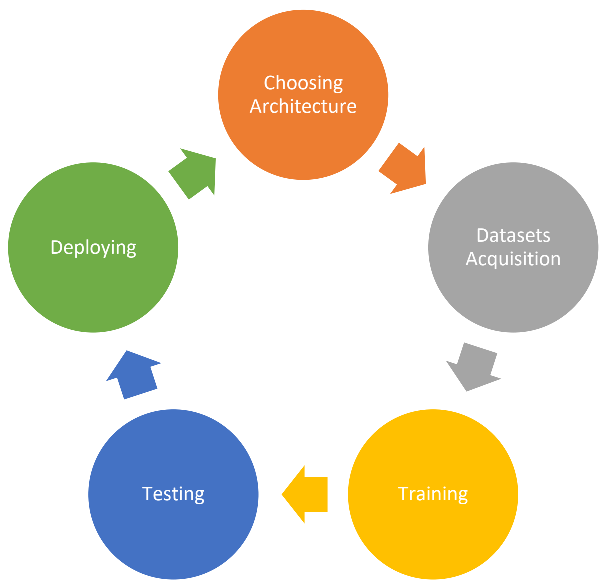


Figure 11-1 First Model Cycle (Agile)

11.1.3 Rest of the models

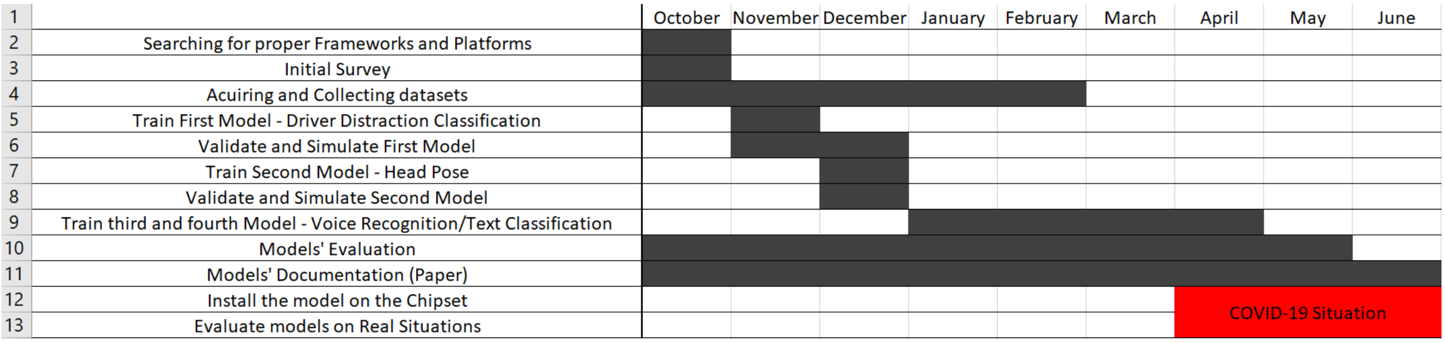
Each team member was assigned to a model to finish and discuss with the rest of the team and mentor. Then we would switch models to gain understanding of the whole system.

11.1.4 Final two months

Final two months had a huge impact because we raised the accuracy of the whole system because of what we learned throughout the year (Whole Accuracy was raised by 9% average). Then we were ready for deploying the system on the Nvidia board but due to the pandemic we couldn't ship the board. So we tested our system on mini CPU refereed in Section 8.

11.2 Planning (Appendix B)

11.2.1 Gantt chart



11.2.2 Funding

To develop the hardware system, we needed funding from organizations and funding teams. We signed to multiple organizations (names won't be added)[11]

Item	Type	Vendor/Source	Unit Cost
1	Jetson Nano Developer Kit	NVIDIA / Amazon	100$
2	Camera	Any (e.g. Arducam)	40$
3	Microphone	Any	35$

11.2.3 Mentorship

As the system software is complicated, we signed to a mentorship program and we were accepted and they were a great help to us through feedback and improving the system.

11.2.4 Competitions

We were ready to get in multiple competitions as it is complementary work in our University Graduation Project Judgement team. And to get recognized in multiple platforms to enhance our future careers.

We had a system proposal and a funding report as follow (Before COVID-19).

11.2.5 Deployment and Training

- Frameworks
  - Python – Pytorch – TensorFlow – Kearas – Deep Speech API.
- Models Training Solutions
  - Local machine with GPU – Google Cloud – Google CoLab.

[11] Report will be blind-reviewed in order to ensure complete fairness.

11.3 Issues (Appendix C)

11.3.1 Datasets

The main issues we encountered were dataset acquisition, we had to search for datasets to fit our system we had in mind. We found datasets from Kaggle and Local car suppliers Companies (issued by proposing the idea) in our country to use.

In the text classification model, we didn't find the data that could match our criteria, so we created our simple dataset as in section 7

11.3.2 Hardware Shipment and Funding

To run the model, we had to ship the board and utilities from abroad (Appendix B)

11.3.3 Models' Training

To train the model we had to use powerful GPUs, luckily, we had good PCs to train the models and we used Google colab

11.3.4 Contingency Plan

For any future risk, we had a contingency plan for any issue we thought we could face.

Scenario	Problem / Future Risk	Recovery
NLP Dataset and Language Model	We couldn't find a specific dataset for our problem of car commands	Creating our own dataset
NLP Models Training	The time for training is very huge almost a month and will require a lot of GPUs	We will use a pre-trained model that is provided by deep speech team
NVIDIA Jetson Board shipment	Not available in our country, can't arrive on time	We will acquire it from Amazon, we will try to borrow one from other people
Chipset Installment	Mic, Cameras and models will need an integrated GPU	Mics and Cameras Interfaces to the NVIDIA Jetson Board
Camera and Mics	Our system needs clean audio and clean Picture Frames	We will acquire Cameras and Mics with a nice performance from Amazon

12 Resources

- Eval Metrics
- How to Add Regularization to Keras Pre-trained Models the Right Way
- Fine-Grained Head Pose Estimation Without Keypoints, Nataniel Ruiz, Eunji Chong, James M. Rehg, V5 13 Apr 2018.
- Deep Speech: Scaling up end-to-end speech recognition, Awni Hannun, Carl Case, Jared Casper, Andrew Y. Ng, V2 19 Dec 2014
- Deep Residual Learning for Image Recognition, Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, 10 Dec 2015
- Driver Distraction European Commission 2015
- https://www.rospa.com/Road-Safety/Advice/Drivers/Distraction
- https://www.sciencedirect.com/book/9780123819840/handbook-of-traffic-psychology
- Driver Distraction A Sociotechnical Systems Approach, Katie J. Parnell, Neville A. Stanton and Katherine L. Plant
- https://www.nhtsa.gov/risky-driving/distracted-driving
- 100 DISTRACTED DRIVING FACTS & STATISTICS FOR 2018
- Facts + Statistics: Distracted driving
- 24 Distracted Driving Statistics & Facts – 2019
- Technology That Can Reduce Driving Distractions and Their Dangers
- AUC Driver Distraction Dataset
- Tuning Language Model
- Keras Text Classification
- EDA Paper