Estimating Safety Parameters of Traffic Scenarios using Probabilistic Programming Languages

Thesis Part-I (CS67101) report to be submitted in partial fulfillment of the requirements for the degree

of

Master of Technology in Computer Science and Engineering

by

Md Laadla
(20CS60R20)

Under the guidance of

Prof. Pabitra Mitra

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Department of Department of Computer Science and Engineering
Indian Institute of Technology,
Kharagpur
India - 721302

CERTIFICATE

This is to certify that we have examined the project report entitled **Estimating Safety Parameters of Traffic Scenarios using Probabilistic Programming Languages**, submitted by **Md Laadla**(Roll Number: *(20CS60R20)*) a postgraduate student of **Department of Computer Science and Engineering** in partial fulfillment for the award of degree of Master of Technology in Computer Science and Engineering. We hereby accord our approval of it as a study carried out and presented in a manner required for its acceptance in partial fulfillment for the Post Graduate Degree for which it has been submitted. The project report has fulfilled all the requirements as per the regulations of the Institute and has reached the standard needed for submission.

Prof. Pabitra Mitra

Department of Computer Science and Engineering
Indian Institute of Technology,
Kharagpur

Place: Kharagpur
Date:

ACKNOWLEDGEMENTS

I express my sincere gratitude to Prof. Pabitra Mitra, Assistant Professor, Computer Science and Engineering Department, Indian Institute of Technology, Kharagpur for his supervision and guidance during the project. His valuable ad- vises, encouragement, suggestions and friendly attitude towards me, during this work helped a long way in bringing this project report to this stage. I am also thankful to him for extending all kind of help and for providing the necessary facilities required during this work.

I express my sincere gratitude to Ashiqur Rahaman (Research Scholar) for providing his valuable guidance, continuous support and encouragement, throughout the course of this work.

I am very much thankful to Prof. Dipanwita Roy Chowdhury, Head, Department of Computer Science and Engineering Department, IIT Kharagpur for providing necessary facilities during the project work.

I am thankful to Faculty Advisor Prof. Soumya Kanti Ghosh and all the faculty members of the department for their valuable suggestions, which helped me improve on this work.

I would like to thank my family for all their love and encouragement. I would like to thank my parents, whose love and guidance are with me in whatever I pursue.

Md Laadla
IIT Kharagpur
Date:

ABSTRACT

Deep Learning models have been widely used in the domain of self-driving cars. One of the major challenges in this domain is generation of real data. We resort to synthetic data generation in this regard. The other important aspect in self-driving cars is safety. Inclusion of AI and ML techniques in self-driven cars open doors for various concerns relating to safety. A lot of works in the AI domain are in transit concerning the safety of these self driven cars. We propose methodologies to leverage the advantages of probabilistic programming languages to infer the distributions of various features extracted from self-driven car simulations to learn the safety of these driving scenarios. Simulation of various cars on predefined synthetically generated roadmap has been done to identify data coming from various sensors which we take as safety features. We resort to Pyro as our Probabilistic Programming Language to learn the inference of the safety features and finally we plot them. Finally, the plot shows the population distribution of the safety features by only considering the sample data.

Keywords: self-driven cars, safety, probabilistic programming, synthetic data.

Contents

List of Figures

Chapter 1

Introduction

In the recent decades, autonomous driving system research has grown in importance, causing significant disruption in the automotive industry. Automotive driving technology has been identified as one of the promising answers to these social and environmental challenges, with demands on reducing traffic accidents, congestion, energy consumption and emissions. By eliminating human error from driving, autonomous vehicles have the potential to save thousands of lives. Autonomous vehicles will also benefit specific groups. People of all ages, as well as those with disabilities who are unable to drive themselves will benefit from mobility thanks to this new technology.

According to the National Crime Records Bureau, Ministry of Road Transport and Highway, Law Commission of India, Global status report on road safety 2013, there is one death every 4 minutes due to road accidents in India. In order to put self-driving cars on Indian roads, we must take this aspect into account. As a result, it is critical to perform a thorough examination of the available data on AV-related accidents and incidents. The most crucial factor to consider with autonomous vehicles is that they be driven safely and do not cause traffic accidents. ML can also be used to analyze data collected from vehicle equipment to ensure that their failure does not result in an accident. Sensor systems, including cameras, LiDAR and Radar, must be kept in top working order; otherwise, a safe journey cannot be guaranteed. ML is already being applied to numerous areas of the technology used in advanced driver-assistance systems (ADAS) despite the fact that autonomous vehicles are still in the

development and testing stages. It also appears to be a factor in future developments.

To cover the gap between model needs and data availability, synthetic data is information generated by computer simulations or algorithms as an alternative to real-world data. Synthetic data is inexpensive to produce and can aid in the creation of AI and Deep Learning models as well as software testing. Synthetic data creation methods create synthetic data that closely resembles sample data while guaranteeing that all of the sample data's significant statistical qualities are reproduced in synthetic data. Experiments in real life are costly. For example, A self-driving car is frequently educated by having it travel hundreds of miles of simulated highways before ever driving on a real one. Data production can also be done with general adversarial networks, which are generative models that generate new data. As a result of these, synthetic data collection has become more accessible and efficient. AI is a particularly strong technology because of its ability to develop itself using synthetic data. The key to improving the quality and quantity of resilient training data for advanced models and simulations is to synthesize data.

The field of probabilistic programming languages(PPLs) have been exploding with research and innovation in recent years. Most of that innovations have come from combining PPLs and deep learning methods to build some ML models that can efficiently handle uncertainty. Probabilistic programming is a tool for statistical modeling. The idea is to borrow lessons from the world of programming languages and apply them to the problems of designing and using statistical models. Diverse training datasets are essential for AI model development, yet real-world data can be lacking. For developing a varied synthetic environment, randomizing circumstances such as lighting, colors and object placement is critical. For more accurate AI models, training data is required. The variances in these artificial worlds reflect the variations in the real world. They show up in real life, where the unexpected and unpredictable happen on a regular basis.

Chapter 2

Literature Survey

This chapter briefly describes various works that has been done in the past relating to usage of AI and ML in self-driven cars. This section has been divided into three subsections, the describing the past works relating to safety in self-driven cars, probabilistic programming languages, and sensor fusion respectively.

2.1 Approaches relating to Safety of Self Driven Cars

Self-driving cars are the ones in which human drivers are never required to take control to safely operate the vehicle. They use sensors and software to control, navigate, and drive the vehicle, and are also known as "driverless" cars. Self-driving cars are predicted to have a transformative impact across numerous industries, accelerating the next wave of technological progress. Although the level of safety required before drivers embrace self-driving cars is unclear, the criterion of being safer than a human driver has become commonplace in discussions about vehicle automation.

Advances in engineering and technology have heightened public interest in the idea of mass-market self-driving vehicles. The degree to which users accept automation is a key factor in whether the technology succeeds or fails. The public's and customers' acceptance of self-driving cars has yet to be determined, but trust — "the belief that

an agent (in this example, a self-driving car) will assist an individual in achieving his or her goals in a circumstance marked by ambiguity and vulnerability"[5].

In the article [8], safety has been defined in terms of risk, epistemic uncertainty, and the harm caused by unfavourable outcomes. The work includes investigations on how certain real-world applications may not be totally amenable to the core premise of modern statistical machine learning: empirical risk reduction, such as cost function selection and the appropriateness of minimising the empirical average training cost.

In the paper [2] published by NVIDIA in 2016, the authors employed a CNN architecture to extract features from the driving frames in their work. The network was trained with supplemented data, which helped the model perform better. The training set was used to create shifted and rotated pictures with corresponding modified steering angles. This method worked effectively in simple real-world circumstances like highway lane-following and driving on flat, obstacle-free courses. Several studies have been conducted to develop more complicated perception-action models to deal with the diverse environments and unpredictable circumstances that are common in urban settings.

Regardless of the empirical definitions and possible interpretations of safety outlined above, the employment of deep learning components in safety-critical systems remains a concern. The ISO 26262 standard for road vehicle functional safety lays out a thorough set of rules for ensuring safety, however it ignores the particular properties of deep learning-based software.

2.2 Approaches to Probabilistic Programming Languages

PPLs (probabilistic programming languages) are domain-specific languages that express probabilistic models and the mechanics for inferring from them. PPL's magic is based on merging probabilistic methods' inference skills with computer languages'

representational power. In recent years, the field of probabilistic programming languages (PPLs) has exploded with research and invention. The majority of these breakthroughs have come from integrating PPLs with deep learning approaches to create neural networks that can deal with uncertainty effectively.

Scenic is a new probabilistic programming language in the field of self-driving cars[3]. The authors offer a new probabilistic programming language for designing and analysing perception systems, particularly those based on machine learning, in their study. Scenic allows us to assign distributions to scene elements and declaratively impose hard and soft constraints on the scene. They demonstrated how Scenic may be used to create synthetic data sets that can be used for deep learning tasks. They have used Scenic to create specialised test sets, improve training set efficacy by stressing difficult cases, and generalise from individual failed cases to broader scenarios suitable for retraining. They claimed to have a great improvement in performance of a car detection neural network (given a fixed training set size) by training on challenging examples created by Scenic, compared to other synthetic data generation methods not based on PPLs. It has been demonstrated that networks trained just on synthetic images from the video game Grand Theft Auto V may attain good performance on actual photographs (GTAV) by creating Scenic scenarios sampler and using it to generate scenes that GTAV converted into photos.

Scenic can generate data of any form (for example, RGB pictures, LIDAR point clouds, or trajectories from dynamical simulations) by interfacing with the right simulator. There are only two steps to this: (1) creating a tiny Scenic library that defines the sorts of objects supported by the simulator as well as the workspace geometry; (2) creating an interface layer that converts Scenic configurations into the simulator's input format.

Another recent work in probabilistic programming language is pyro[1]. Pyro is a probabilistic programming language created by Uber AI. Pyro is a framework for developing advanced probabilistic models in AI research that is built on Python. Pyro combines stochastic variational inference algorithms and probability distributions built on top of PyTorch, a new GPU-accelerated deep learning framework, to

scale to big datasets and high-dimensional models. We'll be adhering to pyro as our probabilistic programming language in this paper.

2.3 Approaches on Sensor Fusion

As we have seen that synthetic data generation helps us to generate data cost effectively and also we can handle uncertainties in these data by using probabilistic approaches. For the purpose of generating synthetic data, we need to simulate our proposed models such that we can extract information from these simulations and hence gather data. The major part of this kind of data generation revolves around the methodologies by which we will measure and quantify the information from these simulations. In particular, for the case of self-driven cars we need to use various sensors for measuring various aspects of the scenarios like velocity of the cars, object detection around cars and roads etc. In this regard we can also investigate sensor fusion, that is fusion of different external sensors, such as a gyroscope and a magnetometer, with in-vehicle sensors, to increase machine learning identification of unsafe driver behavior[4].

Many works have also used sensors embedded on smartphones to identify more data [6]. They have used the gyroscope, accelerometer, GPS, microphone sensors that are usually present in almost every smartphone and finally proposed methodologies to detect three dangerous driving events: speeding, irregular driving direction change, and abnormal speed control.

Other methods include using in-car sensors like ultrasonic sensors, power steering sensor described and this is attached to the embedded system [7]. Using arduino, machine learning algorithms for object detection and ultrasonic sensors, they tried to detect the possibility of an accident.

Chapter 3

Methods

This chapter explains the research problem and its objective, the problem set up, creation of road map, synthetic data generation, sensor fusion and generalised estimation and variational inference model.

3.1 Research Problem/Objectives

Safety has become a great concern in self-driving cars since its introduction to mankind. This work brings out the aspects of predicting safety in self-driven cars. Suppose, we have a geographical knowledge of the road segments of an area. The roads are fixed and only the cars and traffic are dynamic. Studying the daily scenarios of the vehicles along these roads and coming up with some learned knowledge relating to the movements of the vehicles, that ultimately defines the safety of the moving objects on these roads forms the basic foundation of my work(Fig. 3.1). The learnable aspects of these road scenarios should be such that they handle the uncertainties. Diverse training datasets are key for building AI models, but real-world data can fall short. In this work, domain randomization plays an important role in the prediction of these safety features. "Reality is really random" - this proverb holds true for almost all the works in AI and this work is no exception.

After the identification of these safety features and then applying ML models on

these features will help in predicting the safety of various driving scenarios on the fixed road map that we have considered.

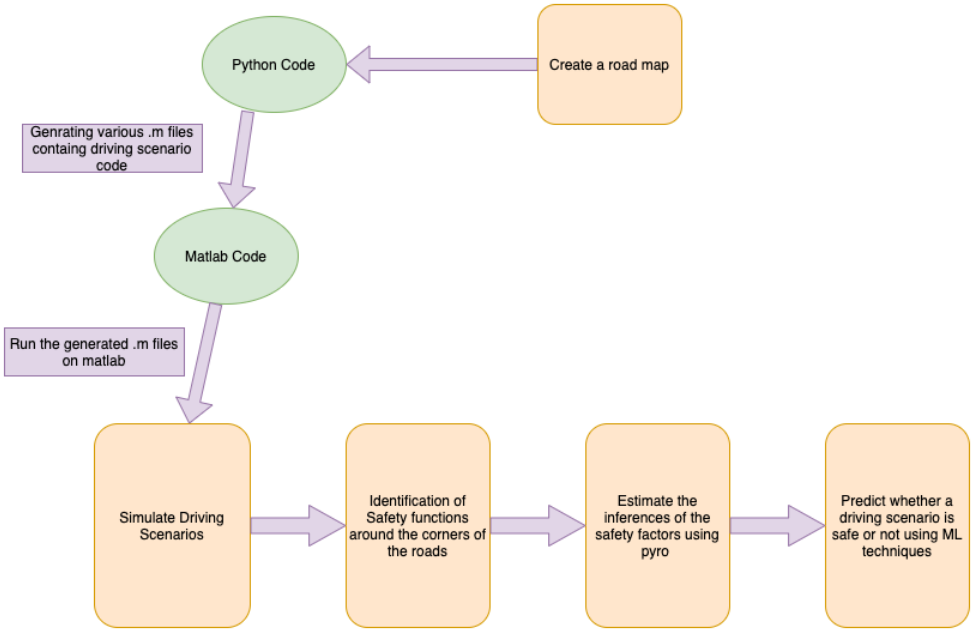


Figure 3.1: General Workflow

3.2 Problem Setup

The first and the foremost requirement for any ML model is data. Generating data for self-driving cars is really expensive and resource intensive. Thus we generated data using artificial means. We have created various traffic scenarios and simulated them on the road maps. The simulation was such that the vehicles on the road follow the Indian system of driving ("always stay to your left while driving forward"). When running these driving scenarios numerous times on the same road map, we try to gather learnable features that best fit our consideration of safety.

3.3 Creation of Road Map

As an early work the road map is considered as a grid adhering to the fact that many of the planned localities like Salt Lake (Kolkata) consists of grid road network. The road consists of two lanes. The coordinates of the roads are pre-decided.

3.4 Data Generation

The entire work is based on synthetic data. We allow the cars to move on these fixed roads randomly. The randomness is such that whenever there is an intersection of roads, a vehicle is free to take any direction along the roads randomly. The speed of the moving vehicles is also decided randomly. For each car moving on the roads we generated various waypoints. Special efforts have been made to keep the vehicles in their respective lanes during their movements and also during changing road segments.

The starting and ending waypoints for each car is chosen randomly. All the way points between the starting and ending are along the road paths. Each intersection of the roads is considered to be a node in a graph. The graph consists of various nodes and the number of nodes is equal to the number of intersections of roads. We have applied depth-first traversal on this graph in order to generate the waypoints. Suppose a car is at a node n, now it can move to all the neighbours of n randomly, if such neighbour exists in the graph.

Every node in the graph is a point in x-y plane. Currently if a car is in a node n represented by (x,y) coordinate, the car has four coordinates representing the neighbours of the node n to choose from:

- (x , y+1) if the car goes right
- (x , y-1) if the car goes left
- (x-1 , y) if the car goes in the same direction but in another node
- (x+1 , y) if the car goes in the reverse direction

Let the new coordinates that a car moves to be x' and y' accordingly. Whenever a car changes a road segment an offset is used to allow the car to change its lane in such a way that makes sense to simulate a real world scenario. For example, if a car goes right from its previous direction then, the offset that we add to or subtract from x' and y' makes sure that the car remains towards the left side of the road segment (following Indian standards). The offset that we have chosen is $\frac{w}{2}$, where w is the width of the road.

Between every waypoint that we have generated, we also generate more waypoints to make sure that the path taken by a car does not change its course to another lane or even outside the road while simulation. For example, if we generate a waypoint say, p and another successive waypoint say q, we generate more points in the line formed by the points p and q.

3.5 Sensor Fusion and Generalised Estimation

The next step is to identify the safety features. To predict whether a driving scenario is safe or not we need some features to train our ML model on. These safety features are extracted from all the simulations performed. Based on sensor fusion we try to identify the unsafe driving behaviour of self driven cars. In particular, we investigated the use of different external sensors, like the accelerometer, gyroscope, magnetometer etc. Sensor fusion means combining data coming from different sensors measuring the same phenomenon (direct fusion), or from sensors measuring different quantities (indirect fusion). In both cases, the action of merging the data is useful to decrease the uncertainty of the observed phenomenon.

According to the Federal Highway administration, More than 50 percent of the combined total of fatal and injury crashes occur at or near intersections. While simulating all the driving scenarios, we were keen to investigate the readings of the sensors around the intersections of the roads. The various attributes like the vehicle velocity, vehicle engine speed, vehicle yaw and vehicle angular velocity, around the intersection, became our area of interest for the selection of safety features.

Handling uncertainties plays a very important role in AI. We can never always explore or have all the possibilities considered in our training data. The reality is always uncertain. In order to mitigate the loss in learning due to insufficient data we tried a probabilistic approach.

For the above mentioned purpose, we tried to take advantage of probabilistic programming languages like pyro. Much of modern machine learning can be cast as approximate inference and expressed succinctly in a language like pyro.

After gathering data from sensors, we tried to use it to estimate the original distribution parameters. If we consider our sensor data as a Gaussian distribution, the traditional way is to simply take the mean and standard deviation. However to handle uncertainties we tried to get the parameters of the full distribution rather than point estimates, as it should generalise our training data.

3.6 Variational Inference

Variational Bayesian methods are a family of techniques for approximating intractable integrals arising in Bayesian inference and machine learning. This is equivalent to solving a Bayesian posterior estimation problem with gradient descent.

Normally, if we try estimate the posterior following Bayesian methods, we need to calculate:

$$P_\theta(\mathbf{z}|\mathbf{x}) = \frac{P_\theta(\mathbf{x}|\mathbf{z})P_\theta(\mathbf{z})}{P_\theta(\mathbf{x})}$$

where, $\mathbf{z}$ is a latent variables and $\mathbf{x}$ is the observed data.

The denominator $P_\theta(\mathbf{x}) = \int P_\theta(\mathbf{x}|\mathbf{z})P_\theta(\mathbf{z})d\mathbf{z}$ translates into an intractable integral in high dimensional spaces. One remedy to get around this problem of not being able to compute the posterior because of the normalising constant or the evidence is to use variational inference. Variational inference suggests to use another distribution

as an approximation of the posterior.

Now, the goal is to simply approximate the intractable posterior distribution with a simpler distribution.

$$P_\theta(\mathbf{z}|\mathbf{x}) \approx Q_\phi(\mathbf{z}|\mathbf{x})$$

.
An optimization procedure here requires a loss function i.e., we need a way to compute the dissimilarity between the ground truth and the predictions using the parameters we are learning. So we use KL divergence.

We make a prior assumption of the type of distribution the posterior should follow. After this assumption we try to tune the parameters to minimise the difference between the actual posterior and the estimated posterior.

We specify the type of distribution the posterior should follow. We then tune the parameters to minimise the difference between the estimated posterior and the actual posterior. The difference measure used is the KL divergence. (KL divergence is not symmetrical).

$$D_{\mathrm{KL}}(Q_\phi \parallel P_\theta) \triangleq \sum_{\mathbf{Z}} Q_\phi(\mathbf{Z}) \log \frac{Q_\phi(\mathbf{Z})}{P_\theta(\mathbf{Z} \mid \mathbf{X})}$$

$$= \mathbb{E}_{Q_\phi(\mathbf{Z})} \left[\log \frac{Q_\phi(\mathbf{z}|\mathbf{x})}{P_\theta(\mathbf{z}|\mathbf{x})} \right]$$

$$= \mathbb{E}_{Q_\phi(\mathbf{Z})} \left[\log Q_\phi(\mathbf{z}|\mathbf{x}) \right] - \mathbb{E}_{Q_\phi(\mathbf{Z})} \left[\log P_\theta(\mathbf{z}|\mathbf{x}) \right]$$

$$= \mathbb{E}_{Q_\phi(\mathbf{Z})} \left[\log Q_\phi(\mathbf{z}|\mathbf{x}) \right] - \mathbb{E}_{Q_\phi(\mathbf{Z})} \left[\log \frac{P_\theta(\mathbf{z}, \mathbf{x})}{P_\theta(\mathbf{x})} \right]$$

$$= \mathbb{E}_{Q_\phi(\mathbf{Z})} \left[\log Q_\phi(\mathbf{z}|\mathbf{x}) \right] - \mathbb{E}_{Q_\phi(\mathbf{Z})} \left[\log P_\theta(\mathbf{z}, \mathbf{x}) \right] + \mathbb{E}_{Q_\phi(\mathbf{Z})} \left[\log P_\theta(\mathbf{x}) \right]$$

$$= \mathbb{E}_{Q_\phi(\mathbf{Z})}\left[\log Q_\phi(\mathbf{z}|\mathbf{x})\right] - \mathbb{E}_{Q_\phi(\mathbf{Z})}\left[\log P_\theta(\mathbf{z},\mathbf{x})\right] + \log P_\theta(\mathbf{x}) \int Q_\phi(\mathbf{z}|\mathbf{x}) d\mathbf{z}$$

$$= \mathbb{E}_{Q_\phi(\mathbf{Z})}\left[\log Q_\phi(\mathbf{z}|\mathbf{x})\right] - \mathbb{E}_{Q_\phi(\mathbf{Z})}\left[\log P_\theta(\mathbf{z},\mathbf{x})\right] + \log P_\theta(\mathbf{x}) \ldots\ldots \textit{Since} \int Q_\phi(\mathbf{z}|\mathbf{x}) d\mathbf{z} = 1$$

Therefore,

$$D_{\mathrm{KL}}(Q_{\phi(\mathbf{Z})} \parallel P_\theta) = \log P_\theta(\mathbf{x}) - \left(\mathbb{E}_{Q_\phi(\mathbf{Z})}\left[\log P_\theta(\mathbf{z},\mathbf{x})\right] - \mathbb{E}_{Q_\phi(\mathbf{Z})}\left[\log Q_\phi(\mathbf{z}|\mathbf{x})\right]\right)$$

$$D_{\mathrm{KL}}(Q_{\phi(\mathbf{Z})} \parallel P_\theta) = \log P_\theta(\mathbf{x}) - \mathbf{ELBO}$$

The estimation starts with a class of approximating distributions. Then it finds the best approximation to the posterior distribution. It minimises the Kullback-Leibler divergence between our approximate distribution and the posterior. This is equivalent to maximising the evidence lower bound (ELBO). When we do variational inference ELBO is the loss function that we use. So, for a fixed θ, as we take steps in ϕ space that increase the ELBO, we decrease the KL divergence between the guide and the posterior, i.e. we move the guide towards the posterior. In the general case we take gradient steps in both θ and ϕ space simultaneously so that the guide and model play chase, with the guide tracking a moving posterior $\log P_\theta(\mathbf{x})$. This now becomes an optimization problem which is best suited to the ML world.

Chapter 4

Result

This chapter specifically discusses the actual work with all the tools and libraries used along with the results and plots of our work.

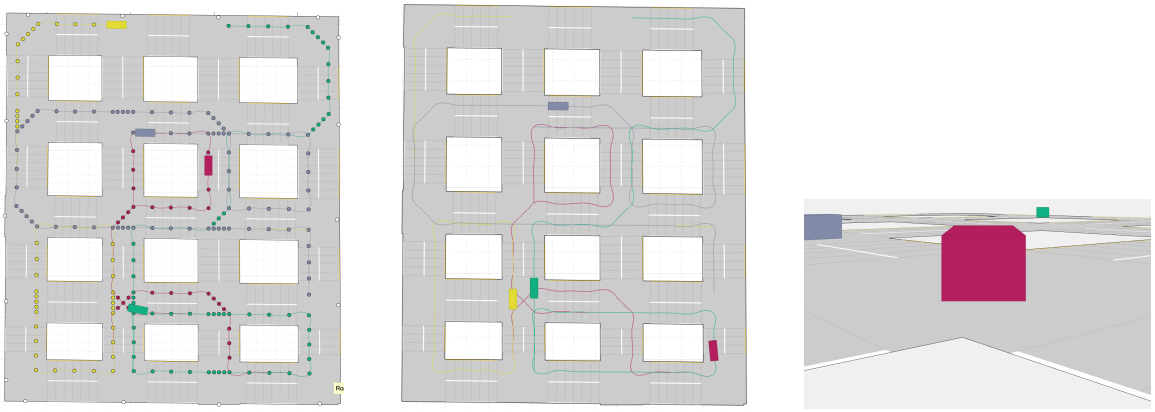
4.1 Experimental Setup

In this work we have simulated 100 driving scenarios. First, we create a road map with fixed coordinates. Then, we have written a code in python which takes in input the coordinates of the road map and generates 100 .m files. Secondly, we run these generated .m files in matlab. The driving scenarios are simulated using matlab SIMULATOR. For each driving scenario simulation we record the readings from various sensors that we use in the simulator. Each *.m* file gathers data from sensors and generates a *.csv* file containing the reading from these sensors. Thirdly, another python code takes in input these *.csv* files, we assume that the sensor readings follow Gaussian distribution and then we use pyro to generate the parameters of the generalised distribution i.e., the distribution of the entire population. At the end we plot the mean and standard deviation of these generalised distributions.

4.2 Generation of Road Maps and the Driving Scenarios

The road map is created in Matlab Driving Scenario Designer Simulator. We have fixed the coordinates of the road network in our entire experimentation. The width of the roads is fixed to be around 10 m in the virtual scenario. The lane width is fixed to 4.925 m. The roads are meant to have only two lanes. No elevated roads are considered in these early simulations. The coordinates of the roads can be found in **appendix**.

We have used python to code the randomised way points of the various cars (ego and other actors). The python code is given input the coordinates of the road map. We create a graph containing the intersection points of each road segment. We have used depth-first traversal for randomised way point generation. We have used 4 cars in total. One is the ego vehicle, and the others are actor vehicles. The simulation is such that we can view the roads and other entities with respect to the ego car. The following figures shows the images of any two random scenarios. Fig. 4.1a, shows the bird eye view of the generated waypoints of all the actors and the ego vehicles of a random scenario say, A. Fig. 4.1b shows the path taken by various cars for the scenario A. Fig. 4.1c shows the ego centric view of the scenario A considered above.

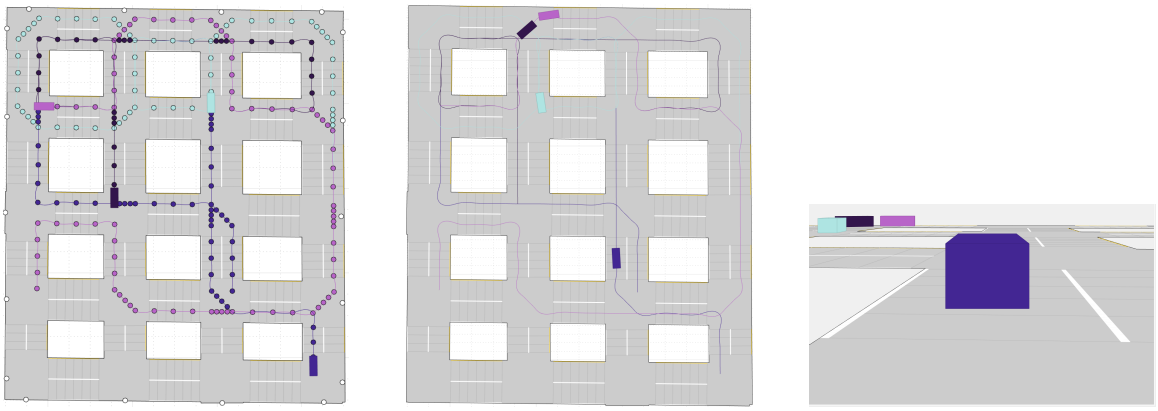


(a) waypoints (b) path (c) ego car view

Figure 4.1: Scenario A

Similarly, Fig. 4.2a, shows the bird eye view of the generated waypoints of all the

actors and the ego vehicles of a random scenario say, B. Fig. 4.2b shows the path taken by various cars for the scenario B. Fig. 4.2c shows the ego centric view of the scenario B considered above.



(a) waypoints (b) path (c) ego car view

Figure 4.2: Scenario B

In the beginning we generated the waypoints only around the end of the road segments. But due to this if the distance between any of the two waypoints is very large then the course of the path changed as such sometimes the car moved outside its lane or sometimes even outside the road segment. So we decided to generate more waypoints between two successive waypoints. The extra points between a line formed by two waypoints is generated using interpolation technique. The function $interp1d$ of the *scipy.interpolate* package has been used.

4.3 Handling Uncertainty of Safety Features using Pyro

4.3.1 The pyro generating model

We tried to fit a Gaussian distribution to our data. The pyro model function takes as input the parameters of the estimated distribution and generates a random sample from the resulting distribution. The parameters of the considered Gaussian distribution come from other distributions. The mean is taken from another Gaussian

distribution. The standard deviation comes from a Gamma distribution.

$$x \sim \mathcal{N}\left(\mu, \sigma^2\right)$$

$$\mu \sim \mathcal{N}\left(\mu_\mu, \mu_{\sigma^2}\right)$$

$$\sigma \sim \text{Gamma}\left(\alpha, \beta\right)$$

The pyro model takes as input the parameters of these two distributions $(\mu_\mu, \mu_\sigma, \alpha, \beta)$. The input is a list of 4 parameters viz the prior mean which consists of two parameters that is the mean and standard deviation of a Gaussian distribution and the prior standard deviation which consists of two parameters from a Gamma distribution. We have used torch.abs to make the distribution parameters always greater than 0. We make the function such that the samples produced by this data model function are enforced to match those from our original sensor data.

4.3.2 The pyro guide

The guide serves as an approximation to the posterior. The guide function represents the family of distribution we want to consider as our posterior distribution, therefore it should be an approximation of the model posterior distribution. In this case we assume a Guassian distribution as the approximating class for the posterior distribution.
The pyro guide must adhere to the following rule:

- The guide function must take the same parameters as the generating model
- The data seen from the model must be valid outputs from the guide function

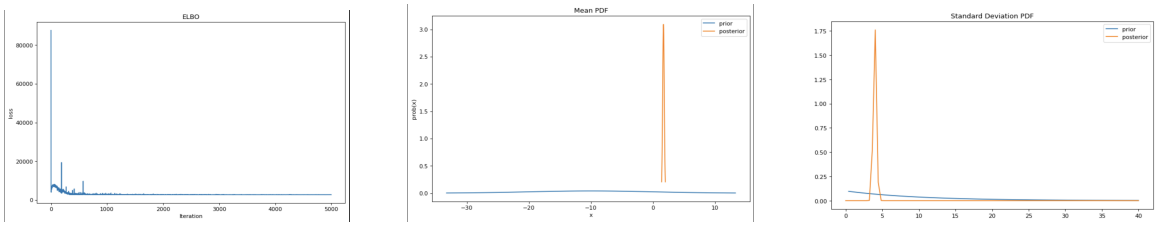
These functions are built with pyro primitives so that they can be used to optimise the KL divergence. The parameters of the guide function are the same as the data generating model. The *pyro.param* statements recall the named parameters from the pyro param store. If no parameter exists with that name it will use the param[.] value passed to it, this happens on the first call only.

We make both learnable parameters that are mean and standard deviation as separate objects in order to optimise the mean and standard deviation of our data separately.

4.3.3 Variational inference descent and the learning paradigm

In our conquest of learning the distribution we have chosen uninformed priors for the mean (Gaussian $\sim \mathcal{N}(0.0, 10.0)$) and the standard deviation ($\sigma \sim$ Gamma $(1.0, 0.1)$). The reason for the same being that we wanted to learn from the data without assuming any significant previous knowledge. Starting with the chosen priors we iterate over our data using the Adam optimizer from the pyro.optim module. The learning rate (lr) is kept as 0.3 and the *betas* are kept as $(0.95, 0.999)$ as the parameters for the optimizer. We have fixed the number of iterations to 5000. At each iteration the SVI pushes our estimated posterior distribution closer to the actual posterior from the data each time. At each step we have stored the parameters so that we could inspect them afterwards. The loss function to optimise is the evidence lower bound. We have used the $pyro.infer.SVI()$ using the functions (generating model and guide) generated above to learn the parameters of the distribution.

In order to check the correct working of the inference model, we have sampled 1000 data from a Gaussian distribution with mean 2 and standard deviation 4. After this we calculated the sample mean as 1.7653922255563976 and sample standard deviation as 3.870407390270422. This confirms that the inference model we have considered works correctly.



(a) Loss (b) mean (c) std

Figure 4.3: Validating the model

The Fig. 4.3 shows the plots of the aforementioned result. The Fig. 4.3a is the loss

with respect to the number of iteration. The Fig. 4.3b shows the prior mean assumption and the estimated mean of the population. The Fig. 4.3c shows the prior standard deviation assumption and the estimated standard deviation of the population.

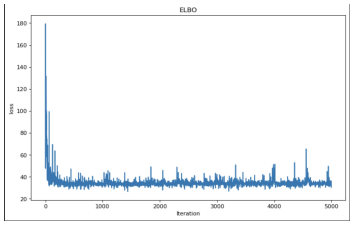
4.4 Distribution Estimation of the Safety Features

As of the current progress of the work we have identified three safety features using sensor fusion in our aforementioned driving scenarios. The functions include:

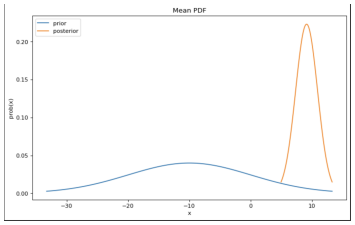
- **Magnitude of the velocity** : From the sensors we have got velocity in x, y and z direction and using them we have calculated the magnitude of the velocity.
- **Yaw angle** : The yaw angle is the angle between a line pointing in the direction the car is moving and the car's x-axis (which is the direction the car is pointed).
- **Angular velocity** : It determines how fast a car is rotating. We define angular velocity as the rate of change of an angle.

All of these are measured around the intersections of the road segments. For each scenario we have taken the average of the measurements of each safety feature around all the intersections of the roads. For example, for the yaw angle, in a driving scenario we will have many yaw angles at the intersections of the roads (since many cars can move through each intersection), we have taken the average of all these yaw angles. This means we have got one averaged yaw angle measurement for each driving scenario. Similar operations has been done for other features.

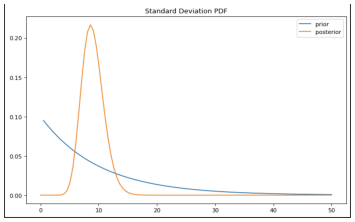
After identifying the aforementioned safety functions, we use pyro to learn the parameters of their distribution using variational inference algorithms. We have assumed that all these features follow a Guassian distribution and also we have assumed that the mean of these Gaussian distributions that they follow also follows a Gaussian distribution (prior) and their standard deviation follows Gamma distribution(prior). The following diagram shows the loss, estimated mean and estimated standard deviation of all the safety features.
The Fig. 4.4 shows the plots of the velocity feature. The Fig. 4.4a is the loss with



(a) Loss



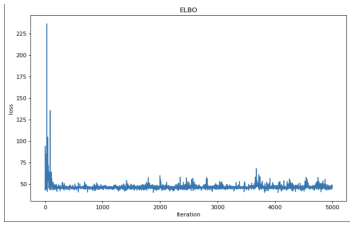
(b) mean



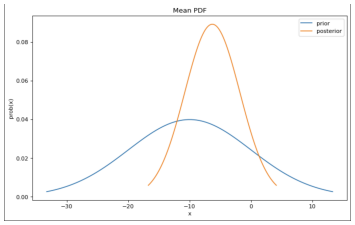
(c) std

Figure 4.4: Velocity

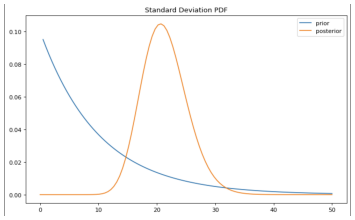
respect to the number of iteration. The Fig. 4.4b shows the prior mean assumption and the estimated mean of the population. The Fig. 4.4c shows the prior standard deviation assumption and the estimated standard deviation of the population.



(a) Loss



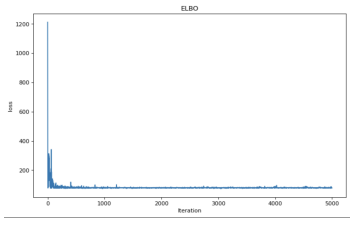
(b) mean



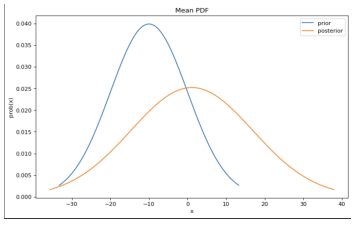
(c) std

Figure 4.5: Yaw Angle

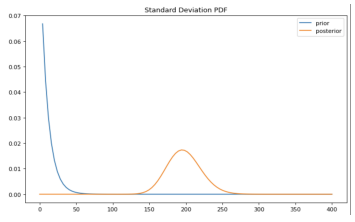
The Fig. 4.5 shows the plots of the yaw angle feature. The Fig. 4.5a is the loss with respect to the number of iteration. The Fig. 4.5b shows the prior mean assumption and the estimated mean of the population. The Fig. 4.5c shows the prior standard deviation assumption and the estimated standard deviation of the population.



(a) Loss



(b) mean



(c) std

Figure 4.6: Angular Velocity

The Fig. 4.6 shows the plots of the angular feature. The Fig. 4.6a is the loss with respect to the number of iteration. The Fig. 4.6b shows the prior mean assumption and the estimated mean of the population. The Fig. 4.6c shows the prior standard deviation assumption and the estimated standard deviation of the population.

Chapter 5

Conclusion and Future Work

Since we are able to find out the various safety features from 100 driving simulations for a small road network, now our next goal is to extend our methodologies for a larger road network with more number of cars and generate 10,000 driving scenarios. Since we were keen at generating synthetic data rather than working on real data which is a quite expensive affair, the greater part of our initial works were quite involved in thinking about data generation. After having done this work, we can now easily extend our model to fit a larger network and generate a large amount of scenarios. The next challenge in our future work is to identify more safety features on which we can apply probabilistic logic using pyro and handle uncertainties in our model, in a similar fashion as we have done for the observed safety factors as discussed in our work. Then we want to find out some methodologies to predict the safety outcome of the observed safety features and also the ones we will be incorporating in the future. We will end our work by training some ML models on these safety factors and predict whether a driving scenario is safe or not.

Bibliography

[1] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. Szerlip, P. Horsfall, and N. D. Goodman. Pyro: Deep universal probabilistic programming. *The Journal of Machine Learning Research*, 20(1):973–978, 2019.

[2] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[3] D. J. Fremont, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia. Scenic: a language for scenario specification and scene generation. *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, Jun 2019.

[4] E. Lattanzi, G. Castellucci, and V. Freschi. Improving machine learning identification of unsafe driver behavior by means of sensor fusion. *Applied Sciences*, 10(18):6417, 2020.

[5] J. D. Lee and K. A. See. Trust in automation: Designing for appropriate reliance. *Human factors*, 46(1):50–80, 2004.

[6] C. Ma, X. Dai, J. Zhu, N. Liu, H. Sun, and M. Liu. Drivingsense: Dangerous driving behavior identification based on smartphone autocalibration. *Mobile Information Systems*, 2017, 2017.

[7] P. G. Namita Surkar, S. S. Rasna Karemore, and S. Jaiswal. Sensor fusion for obstacle detection using machine learning:review. *International Journal of Future Generation Communication and Networking*, 13:1437–1440, 2020.

[8] K. R. Varshney. Engineering safety in machine learning. In *2016 Information Theory and Applications Workshop (ITA)*, pages 1–5. IEEE, 2016.