Capstone Project

AuE-8930 - Deep Learning: Applications in Engineering

Chinmay Vilas Samak
Tanmay Vilas Samak
Vasanth Seethapathi

Spring 2023

This work is an outcome of the course AuE-8930 "Deep Learning: Applications in Engineering" (Spring 2023) at Clemson University International Center for Automotive Research (CU-ICAR) handled by Dr. Rahul Rai, Dean's Distinguished Professor at CU-ICAR.

This report was submitted to the Department of Automotive Engineering, CU-ICAR as a part of the course AuE-8930 " Deep Learning: Applications in Engineering ".

DISCLAIMER

I certify that all the work and writing that I contributed to here is my own and not acquired from external sources. I have cited sources appropriately and paraphrased correctly. I have not shared my writing with other students (for individual assignments) and other students outside my group (for group project), nor have I acquired any written portion of this document from past or present students.

Authors

ABSTRACT

Motion models are crucial for simulation and state estimation purposes and can be either developed using traditional or learning-based methods. On one hand, formal methods such as first-principles modeling are interpretable and explainable but require extensive system identification for parameter tuning and may result in computationally complex models that may not be solved in real time. On the other hand, simple data-driven methods such as end-to-end deep learning can decently capture the dynamics with low-latency inference, but the interpretability and explainability of these models may be questionable. This work investigates the application of probabilistic Bayesian deep learning methods to help bridge the gap between formal and data-driven approaches to develop computationally light-weight surrogate models that are generalizable, interpretable, and explainable. Furthermore, this work proposes a hybrid model architecture to leverage the combined benefits of both point estimate and probabilistic Bayesian neural networks. Particularly, we apply the proposed approach to "learn" a reduced-order (i.e., surrogate) dynamics model of an Ackermann-steered vehicle under Markov assumption given its low-level control inputs such as throttle, brake and steering, as well as rate of change of state variables, and comparatively show how this approach not only helps fit and generalize over the data better, but also quantify the aleatoric and epistemic uncertainties with its predictions thereby imbibing explainability into these models.

KEYWORDS

Deep Learning, Neural Networks, Probability Theory, Bayesian Inference, Vehicle Dynamics, Data-Driven Modeling

DATASET

Publicly available at https://github.com/Tinker-Twins/AutoDRIVE-Nigel-Dataset

CODEBASE

Subject to academic integrity, available at https://github.com/Tinker-Twins/Deep-Learning-Applications-in-Engineering/tree/main/Final-Project

ACKNOWLEDGEMENT

At the commencement of this report, we would like to add a few words of appreciation for all those who have been a part of this project; directly or indirectly.

We would like to express our immense gratitude towards Department of Automotive Engineering, Clemson University International Center for Automotive Research (CU-ICAR) for offering this course and for providing us with an excellent atmosphere for working on this project.

We would also like to express our deep and sincere gratitude to our faculty Dr. Rahul Rai and teaching assistant Siqi Zeng for their valuable guidance, consistent encouragement and timely help. We are also thankful of our peers in the course, who engaged in thoughtful conversations and discussions, which led to cross-pollination of ideas.

Finally, we are grateful to all the sources of information without which this project would be incomplete. It is due to their efforts and research that our report is more accurate and convincing.

Authors

TABLE OF CONTENTS

1. INTRODUCTION

1.1. Motivation

Real-time multi-step simulation of complex nonlinear first-principles multi-body models such as those of automotive systems can be extremely compute-intensive in nature. As such, with limited computational resources, these models often take up a lot of time to be solved. However, these models are highly accurate, explainable, interpretable and most importantly can generalize over all the dynamic operating points or conditions.

Simplified physics-based models such as kinematic models reduce the computation burden to a great extent, however these cannot generalize over the entire dynamic operating envelope (e.g. these often fail to capture vehicle behavior over speeds of 40-50 mph).

As a mediator, deep neural networks can be used to implicitly learn the vehicle dynamics characteristics under a broad operating envelope in a data-driven setting and once the trained model is available, just run inference over it to relax the computational burden. Although this seems promising, one of the key issues with this approach is that being completely black-box models, they lack the interpretability and explainability in terms of their predictions. Moreover, these models have no way of knowing how good their predictions are.

Thus, going a step further, we can leverage rich knowledge of probability theory and Bayesian inference to build probabilistic models with the ability to quantify the uncertainty in their predictions. This way, we also have light-weight models that can generalize over a rather broad dynamic operating envelope without any simplification assumptions (as in case of several first-principles models) and can offer explainability by quantifying the uncertainty in their predictions.

1.2 Objectives

The primary objective of this project was to formulate and apply a probabilistic Bayesian deep learning framework to the problem of vehicle dynamics identification. Particularly we followed the Markov assumption on state propagation and therefore chose the low-level vehicle control inputs (throttle and steering angle) as features and the rate of change of the state variables (linear and angular velocities) to solve a regression problem. This assumption dictates that only a single-step control input affects and explains the system dynamics to the system as opposed to a series of previous control inputs also influencing the system dynamics.

Due to unavailability of openly accessible vehicle dynamics datasets on standard parameterized benchmark maneuvers exciting different aspects of roll- pitch- and yaw-plane vehicle dynamics, the aforementioned primary objective demanded data collection and model training as a set of secondary objectives. The data collection and preprocessing aspect included shortlisting the limits and gradations of control inputs applied to the vehicle, choice of state and rate variables to be measured or estimated, automating the collection of dynamical dataset of an Ackerman-steered vehicle, as well as post-processing the dataset to extract required features and labels for training. The model training and hyperparameter

tuning aspect included definition of the architecture of probabilistic Bayesian neural network, variability analysis on the hyperparameter space, and finally training and saving the model based on optimal set of hyperparameters.

The tertiary objective of this project was to perform model inference (forward-simulation of the trained model by applying timeseries control inputs) and analysis (benchmarking the model against ground-truth and comment on its validity including uncertainty quantification).

1.3 Applications

Following is a brief summary of the potential applications and scope of this project:

- Forward simulation for state propagation.
- Replace hardware sensor by software model for reducing production cost.
- State-estimation in coverage denied areas or in sensor malfunction conditions.
- Redundant variable measurement for robust state-estimation.
- Predicting anomalies in sensor measurements and fault-tolerant control.

2. PROJECT MANAGEMENT

2.1. Project Tools

Following is a brief summary of the tools (**Figure 1**) that went into designing and implementing this project:

- **Vehicle:** Nigel
- **Simulation:** AutoDRIVE Simulator
- **Programming:** Python
- **Framework:** TensorFlow 2

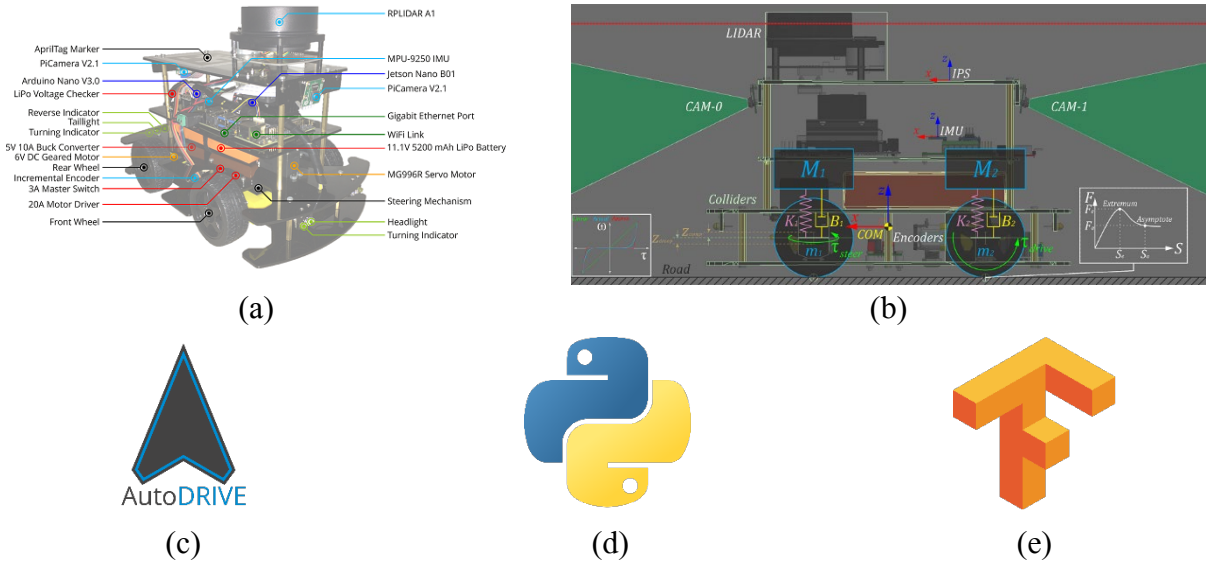


Figure 1. *Project tools: (a) Nigel vehicle; (b) vehicle dynamics model in AutoDRIVE Simulator; (c) AutoDRIVE Ecosystem for data collection; (d) Python programming language; and (e) TensorFlow 2 framework.*

The vehicle used for the project was AutoDRIVE Ecosystem's native 1:14 scale vehicle called Nigel (**Figure 1**). This open-source vehicle adopts car-like constrained actuation with an Ackerman steering mechanism. It is equipped with a wide array of sensors including throttle and steering sensors (actuator feedbacks), 1920 CPR incremental encoders (wheel rotation/velocity), a 3-axis indoor-positioning system (IPS) using fiducial markers (mm/cm-level accurate small-scale positioning analogous to m-level accurate full-scale GNSS), 9-axis IMU (raw inertial data and calibrated AHRS data using Madgwick/Mahony filter), two 62.2° FOV cameras with 3.04 mm focal length (front and/or rear RGB frames) and a 7-10 Hz, 360° FOV LIDAR with 12 m range and 1° resolution (2D laser scan).

Following are some of the important vehicle parameters:

- Wheelbase (l): 0.1415 m
- Track width (w): 0.1530 m
- Throttle limit: 1.0000 norm%
- Steering limit: 0.5236 rad
- Linear velocity limit: 0.2670 m/s
- Angular velocity limit: 0.8051 rad/s

For data collection, we made use of the AutoDRIVE Simulator, a high-fidelity simulation system for autonomy-oriented applications. The simulator has been benchmarked against real-world counterpart (AutoDRIVE Testbed) for perception and dynamics reliability. It also offers a user-friendly data recording functionality, which logs time-synchronized data at the specified target sampling rate.

In terms of the middleware framework used for development of the probabilistic Bayesian deep neural networks (PBNN), we exploited TensorFlow 2 and TensorFlow Probability libraries. The programming for automating data collection, data analysis and pre-processing, defining deep neural network (DNN) models, hyperparameter tuning, DNN training, analyzing DNN performance, etc. was implemented using Python3 (an interpreted language) to ensure code readability and easy integration with modern deep learning libraries.

2.2. Work Breakdown Structure

For our convenience, we divided the project into 3 phases:

- **Phase 1: Data Collection:**
  - Shortlist control inputs and states to be measured/estimated
  - Collect dynamical dataset of an Ackerman-steered vehicle
  - Pre-process the dataset to get required features and labels
- **Phase 2: Model Training:**
  - Define architecture of Bayesian neural network
  - Define/tune hyperparameters
  - Train the model and analyze its performance
- **Phase 3: Model Inference:**
  - Forward-simulate the trained model by providing control inputs to it
  - Benchmark the model against ground-truth and comment on its validity

2.3. Project Timeline

As mentioned earlier, we split this project into three phases viz. data collection, model training, and model inference. Additionally, logistical tasks such as continuous documentation, preparation of slide deck, presentation rehearsals, report writing, etc. had to be accounted for in the overall project plan and strictly followed to ensure successful completion of the project.

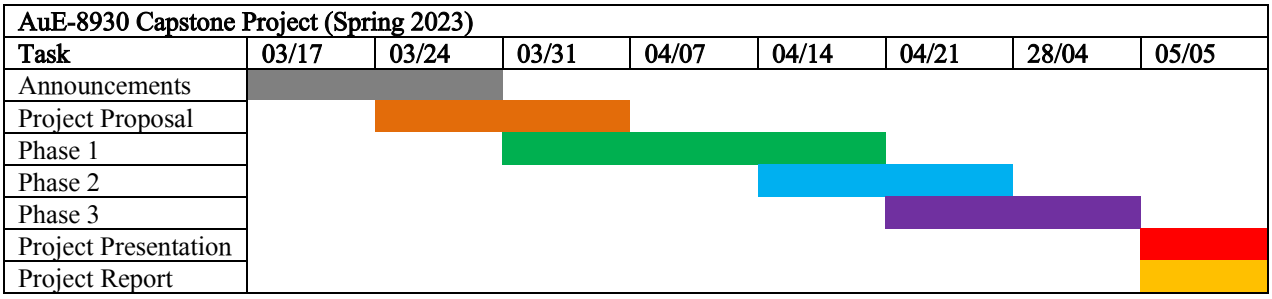


Figure 2. *Project timeline Gantt chart.*

Given the overall timeline from 03/17/2023 (with the beginning of initial announcements regarding the capstone project) to 05/05/2023 (end of Spring 2023 semester), the milestones and deliverables of the project were planned as indicated in **Figure 2**, which depicts the Gantt chart. It is to be noted that things such as known parallel commitments were considered while preparing the timeline so as to enable practical adherence rather than coming up with super tight but highly unrealistic timelines.

2.4. Responsibility Assignment

We divided the project into a set of tasks and assigned a subset of the team members with a primary and secondary responsibility of accomplishing each task as per the planned deadlines. Upon successful implementation of the project (albeit barebone), we all came together for hyperparameter tuning, overall optimization, code cleanup and documentation.

- **Data collection and preprocessing:**
  - Tanmay
  - Vasanth
- **Problem formulation:**
  - Chinmay
  - Tanmay
- **Implementation:**
  - Tanmay
  - Chinmay
- **Hyperparameter tuning and overall optimization:**
  - Everyone
- **Presentation and documentation:**
  - Everyone

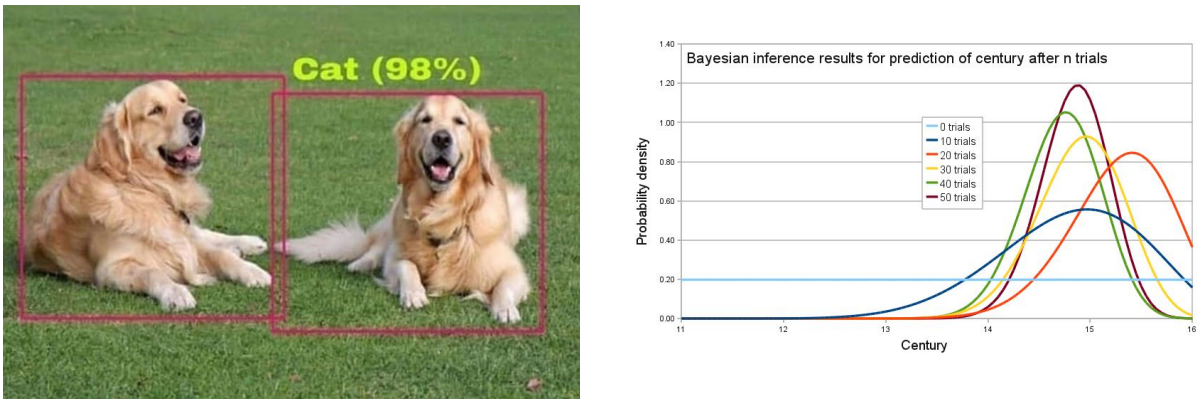
It is to be noted that “responsibility” does not directly indicate “contribution”. The team members have no conflict of interest to declare.

3. BACKGROUND

3.1. Preliminary

The objective of this project is to apply the concept of probabilistic Bayesian deep neural networks to capture or implicitly learn the system dynamics of an Ackermann-steered vehicle under the Markov assumption.

Now, although this problem can be tackled in a number of ways, one of them being using point estimate neural networks, the following is our justification for applying this methodology to the vehicle dynamics problem innovatively.



(a) (b)

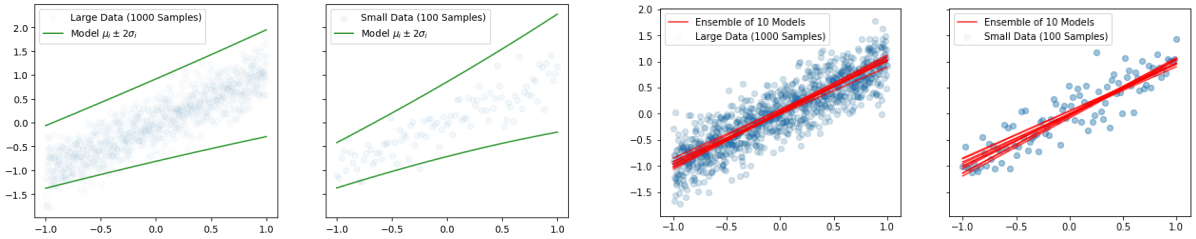
Figure 3. *Motivation to move from point estimate models to probabilistic models: (a) overconfident point estimate models [Source: Reddit]; and (b) Bayesian inference [Source: Wikipedia].*

One of the biggest problems associated with point estimate models (i.e. standard neural network models) is that being completely black-box models, they lack the interpretability and explainability in terms of their predictions. Moreover, these models have no way of knowing how good their predictions are (especially on input data that the model was never exposed while training). As a result, the model can be very confident about its prediction, but be very incorrect at the same time (**Figure 3**).

This is exactly where probability theory and Bayesian inference come in handy. So, now instead of working with point estimates, we are working with probability distributions and sampling from these distributions stochastically. More particularly, we start with a certain prior belief about our estimates, then feed or sample new data points and then use the two to come up with a posterior prediction pertaining to the estimates (**Figure 3**). This recursive estimation method can be easily proven to converge with a reduction in uncertainty after each iteration. However, one small issue with direct application of this method to our problem is that the exact posterior distribution may not exist or may be just too expensive to compute. For this reason, we work with the variational posterior as opposed to the exact posterior and try to minimize the Kullback Leibler (KL) divergence between the two in addition to minimizing the negative log-likelihood of the observed data (or certain loss function making an assumption on the underlying probability distribution of the observed data). What this essentially allows the model is to quantify the uncertainty in its predictions.

3.2. Uncertainty Quantification

Aleatoric (a.k.a. statistical) uncertainty (**Figure 4**) captures noise inherent in the observed data samples. This could be for example sensor noise or motion noise, resulting in uncertainty which cannot be reduced even if more data were to be collected, it can only be quantified (the only way of reducing this type of uncertainty is to apply filtering and preprocessing techniques to the data). This type of uncertainty can further be classified as either homoscedastic or heteroscedastic wherein the former assumes constant observation noise for every input sample, while the latter assumes that observation noise can vary with the input sample. Although homoscedasticity is usually assumed, heteroscedasticity can be particularly useful in cases where parts of the observation space might have higher noise levels than others.



(a) (b)

Figure 4. *Uncertainty quantification: (a) aleatoric uncertainty; and (b) epistemic uncertainty.*

Epistemic (a.k.a. systematic) uncertainty (**Figure 4**) on the other hand accounts for uncertainty in the model parameters and therefore its predictions. This uncertainty captures insufficiency or ignorance/negligence about certain aspects in the collected data. This uncertainty can be explained away and reduced given enough data and is often referred to as model uncertainty.

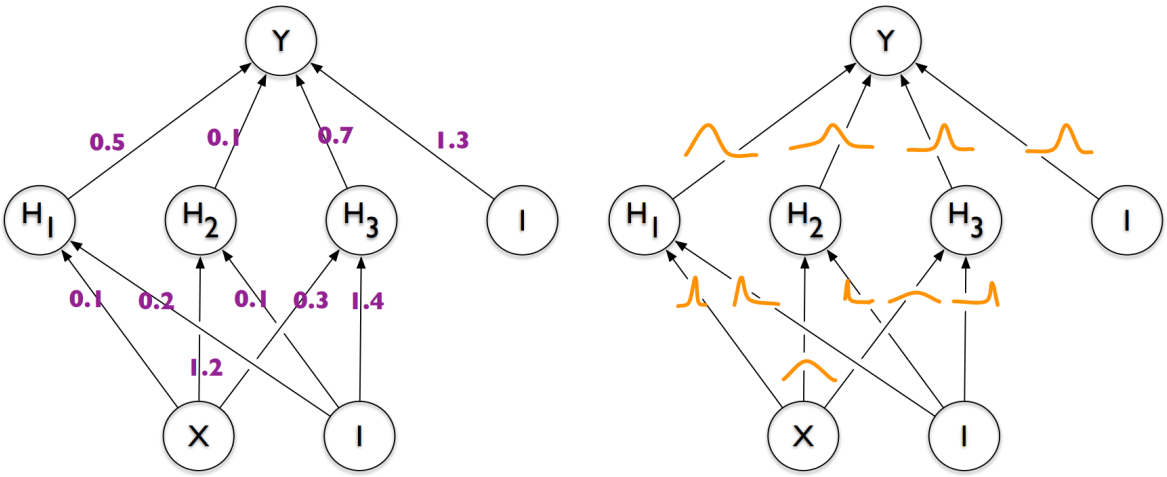
These uncertainties are formalized as probability distributions over either the model parameters (for epistemic uncertainty), or model outputs (for aleatoric uncertainty). Epistemic uncertainty is modeled by placing a prior distribution over a model's weights, and then trying to capture how much these weights vary given some data (i.e. posterior distribution). Aleatoric uncertainty on the other hand is modeled by placing a distribution over the output of the model.

3.3. Probabilistic Deep Learning

This section highlights the key differences or features of various probabilistic deep learning frameworks ranging from completely deterministic standard (point-estimate) neural networks all the way up to the probabilistic Bayesian neural networks, which are of utmost importance for this project (**Figure 5**).

In case of standard or point-estimate neural networks (PENNs), given an input, the model predicts a point-estimate of the output by learning the parameters $\boldsymbol{\theta}$, where $\boldsymbol{w}, \boldsymbol{b} \in \boldsymbol{\theta}$. In case of probabilistic neural networks (PNNs), given an input, the model predicts the likelihood of output based on the underlaying assumption on the probability distribution $\sim \boldsymbol{\theta}$, where $\boldsymbol{w}, \boldsymbol{b} \in \boldsymbol{\theta}$ which helps quantify the aleatoric uncertainty in data. In case of Bayesian neural networks (BNNs), given an input, the model predicts output with certainty by learning (in case of a Gaussian distribution) $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ where $w_i, b_i \sim \mathcal{N}(\mu_i, \sigma_i)$ which helps quantify the epistemic uncertainty. Finally, the probabilistic Bayesian neural networks (PBNNs) can be thought of as

a combination of PNN and BNN unified into one model which helps quantify the aleatoric, epistemic, and potentially even the functional uncertainties.



(a) (b)

Figure 5. *Mechanics of: (a) point estimate neural networks; and (b) Bayesian neural networks [Source: Charles, et al.].*

That being said, it is necessary to weigh the pros and cons of these probabilistic models. On the positive side, these models are more robust and generalizable and can quantify the uncertainty in their predictive output. However, on the negative side, these models can be more complicated to train (requiring knowledge of probability and statistics specific to the problem at hand), and can be slower to converge and require more data (since we are dealing with probability distributions instead of single values).

3.4. Mechanics of Bayesian Deep Learning

BNNs, given an input, predict output with certainty. In other words, they learn the parameters of a probability distribution function (PDF) $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ from which their parameters (weights and biases) are sampled $w_i,\ b_i \sim \mathcal{N}(\mu_i, \sigma_i)$.

Consider a BNN Model: $F_\theta()$ provided with an input vector: $\boldsymbol{x}$, and producing an output: $\hat{\boldsymbol{y}}$, trained on a dataset: $\boldsymbol{D} = (x_i,\ y_i)_{i=1}^{n}$ with a certain loss function: $\mathcal{L}()$. Let the BNN have weights: $\boldsymbol{w}$, bias: b and activation function: $g()$.

Then, the objective of the BNN is to learn $\{\mu_i,\ \sigma_i\}$ such that its parameters when sampled from a normal distribution parameterized by these values $w_i,\ b_i \sim \mathcal{N}(\mu_i, \sigma_i)$ will produce the desired output(s): $\hat{\boldsymbol{y}} = g(\boldsymbol{w}^T\boldsymbol{x} + b) = g(w_1x_1 + w_2x_2 + w_3x_3 + b)$, and this idea can be easily extended to subsequent layers in deep neural networks.

In theory, the training of BNN can be thought of as maximizing the log likelihood of predictions over a given dataset in conjunction with the KL-divergence between true and variational posterior:

$$\mu^*,\ \Sigma^* = \underset{\mu,\ \Sigma}{\operatorname{argmax}} \sum_{(x_i,\ y_i)\ \in\ D} \log[p(y_i|x_i,\ \theta)] - KL[p(\theta),\ p(\theta_0)]$$

where, $\theta \sim \mathcal{N}(\mu, \Sigma)$ and $\theta_0 \sim \mathcal{N}(0, \mathrm{I})$

However, practically, this is achieved by minimizing a loss function which quantifies the "error" in predictions w.r.t. data-truth, in conjunction with the KL-divergence between true and variational posterior:

$$\mu^*, \Sigma^* = \underset{\mu, \Sigma}{\operatorname{argmin}} \sum_{(x_i, y_i) \in D} \mathcal{L}(F_\theta(x_i), y_i) + KL[p(\theta), p(\theta_0)]$$

The prediction step can be theoretically understood as integrating the predictions of the BNN over the entire PDF, given the optimal parameters $\theta^* \sim \mathcal{N}(\mu^*, \Sigma^*)$:

$$p(\hat{y} \mid \hat{x}, D) = \int p(\hat{y} \mid \hat{x}, \theta^*)\, p(\theta^* \mid D)\, d\theta$$

However, practically this is just a forward pass of the neural network $\hat{y} = F_{\theta^*}(\hat{x})$ averaged over multiple instances:

$$\hat{y} = \frac{1}{K} \sum_{k=1}^{K} F_{\theta_k^*}(\hat{x})$$

During the back-propagation step, BNNs face a challenge of intractability since derivatives of the parameters being learnt (i.e. derivatives of distributions) must be calculated:

$$p(\hat{y} \mid \hat{x}, D) = \int p(\hat{y} \mid \hat{x}, \theta^*)\, p(\theta^* \mid D)\, d\theta$$

A work-around solution to this problem is the local re-parameterization trick, which "moves" the parameters to be learnt (μ, σ in case of a Gaussian distribution), out of the distribution function for any weight w.

We know that $\theta = (\mu, \sigma)$. Now let $\epsilon \sim \mathcal{N}(0, 1)$, i.e. a variable that follows standard normal distribution. We can then express the weights as a linear combination of the normal distribution parameters (μ, σ) over which it was being sampled from: $f(\epsilon) = w = \mu + \sigma \cdot \epsilon$

We can now compute derivatives of weights and biases w.r.t. this surrogate function $f(\epsilon)$ and update the parameters to eventually get the optimal parameters $\theta^* = (\mu^*, \sigma^*)$:

$$\Delta\mu = \frac{\partial f}{\partial w} + \frac{\partial f}{\partial \mu}$$

$$\Delta\sigma = \frac{\partial f}{\partial w}\frac{\epsilon}{\sigma} + \frac{\partial f}{\partial \sigma}$$

$$\mu \coloneqq \mu - \alpha\Delta\mu$$

$$\sigma \coloneqq \sigma - \alpha\Delta\sigma$$

4. PRELIMINARY EXPERIMENTS

4.1. Linear Regression

Upon successful completion of problem formulation stage, we conducted several preliminary experiments with probabilistic Bayesian neural networks in order to analyze the behavior of these models and robustify our understanding of these models.

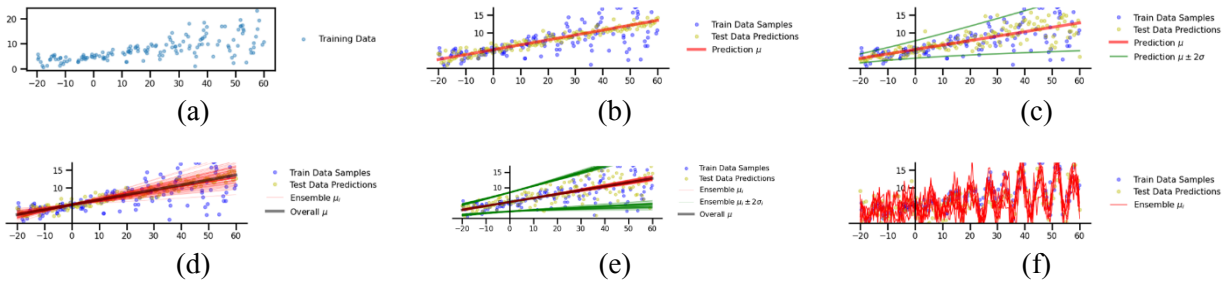
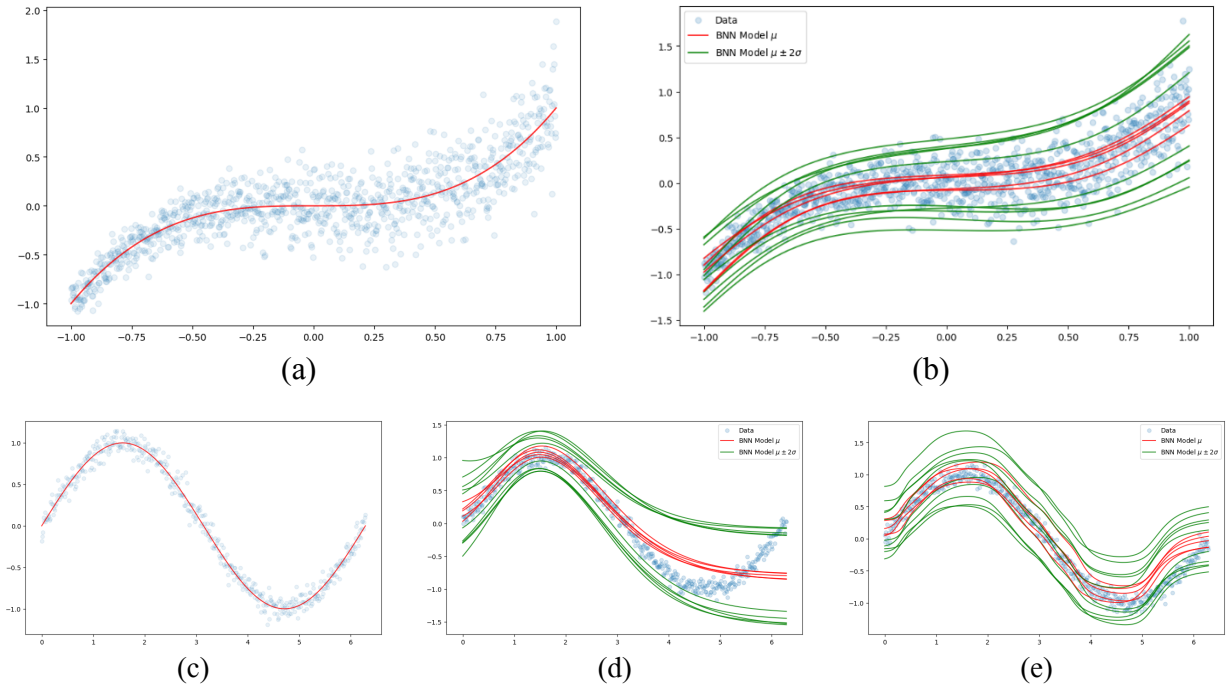


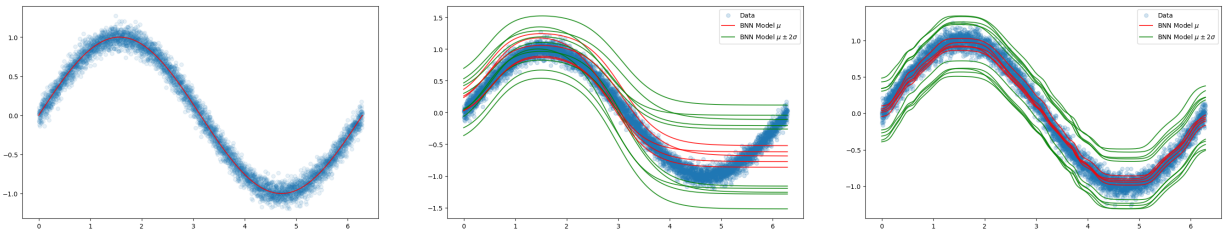
Figure 6. *Linear regression results: (a) data; (b) prediction without uncertainty quantification; (c) prediction with aleatoric uncertainty quantification; (d) prediction with epistemic uncertainty quantification; (e) prediction with aleatoric and epistemic uncertainty quantification; and (f) prediction with functional uncertainty quantification.*

From the linear regression experiments (**Figure 6**), it is evident that the model is very well able to fit the data, but more importantly, also able to quantify the aleatoric uncertainty, epistemic uncertainty, combination of both aleatoric as well as epistemic uncertainties, and also the functional uncertainty.

4.2. Non-Linear Regression

Once satisfactory results were obtained from the linear regression experiments, we explored the problem of nonlinear regression (**Figure 7**), predominantly because the vehicle dynamics problem is inherently nonlinear in nature.





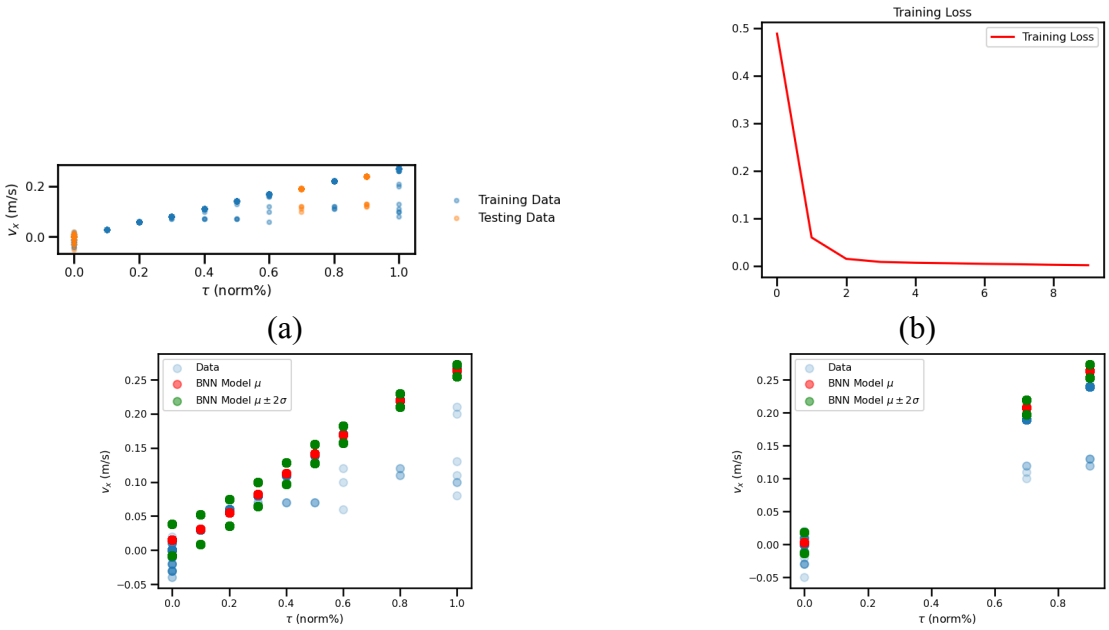
(f) (g) (h)

Figure 7. *Non-linear regression results: (a) cubic function data; (b) simple model predictions on cubic function data; (c) sinusoidal function with less data; (d) simple model predictions on sinusoidal function with less data; (e) complex model predictions on sinusoidal function with less data; (f) sinusoidal function with more data; (g) simple model predictions on sinusoidal function with more data; and (h) complex model predictions on sinusoidal function with more data.*

Particularly, we performed regression on cubic as well as sinusoidal data with variability in number of data samples and model architectures. One of the significant findings was that the performance of the model varies greatly with both quality and quantity of the data as well as the underlying architecture of the model itself.

4.3. 1D Vehicle Dynamics Regression

Upon successful preliminary experiments on linear and nonlinear regression with simplistic synthetic data, we explored a one-dimensional vehicle dynamics problem (**Figure 8**) concerning the decoupled longitudinal dynamics (although the actual vehicle dynamics has coupled longitudinal and lateral components). The results clearly indicate a good fit on training as well as testing datasets (with data samples pertaining to certain control inputs being completely hidden from the model during training) as well as quantification of uncertainty.



(a) (b) (c) (d)

Figure 8. *1D vehicle dynamics regression results: (a) training and testing data; (b) training loss curve; (c) predictions on training dataset; and (d) predictions on testing dataset.*

5. DATA COLLECTION

5.1. Data Collection Platform

We employed AutoDRIVE Ecosystem for collecting the vehicle dynamics dataset. Particularly, we collected the data of AutoDRIVE's scaled vehicle (called Nigel) using the high-fidelity AutoDRIVE Simulator (**Figure 9**). Our initial experiments suggested that this task would demand a lot of data collection/filtering across various operating conditions that the vehicle can undergo. Consequently, we exploited this physically accurate and graphically realistic simulation tool to collect time-synchronized data in controlled conditions.

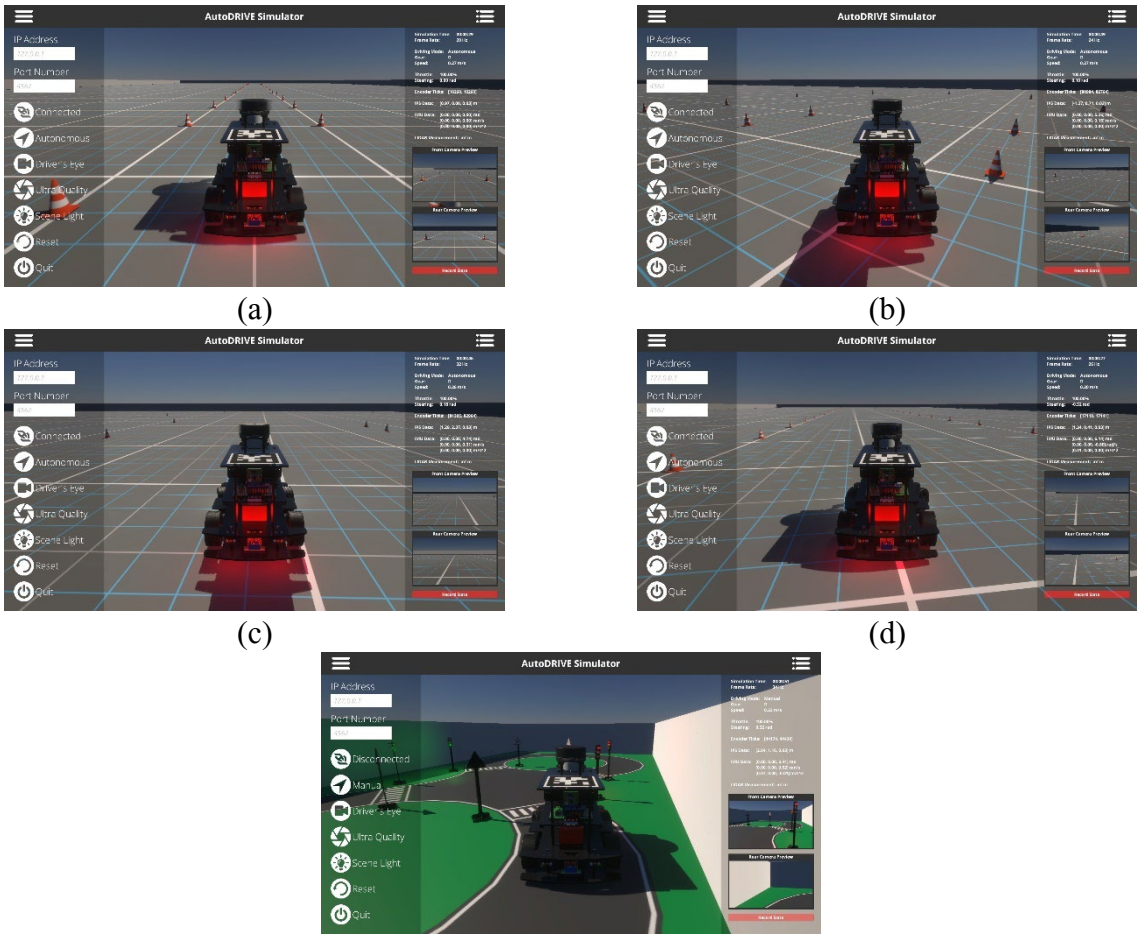


Figure 9. *AutoDRIVE Simulator for data collection: (a) straight maneuver in Testing School scenario; (b) skidpad maneuver in Testing School scenario; (c) fishhook maneuver in Testing School scenario; (d) slalom maneuver in Testing School scenario; and (e) eight maneuver in Driving School scenario.*

Following is a brief summary of the notation used henceforth:

- **Vehicle Controls:** The low-level control inputs of the vehicle are throttle (τ) and steering (δ). Note that negative throttle indicates reverse driving and zero throttle indicates braking condition.
- **Vehicle States:** The most prominent vehicle states being considered are the positional coordinates along X-axis (x) and Y-axis (y) as well as the orientation about Z-axis, i.e. yaw angle (θ). Apart from these, the rate variables including longitudinal forward velocity (v) and angular velocity (ω) of the vehicle are also being considered.

5.2. Data Collection Process

The data collection was partially automated through an `open_loop_controller.py` script, which makes use of AutoDRIVE Devkit's Python API and is capable of selecting a maneuver (`straight`, `skidpad`, `fishhook`, `slalom`) and its direction (`ccw`, `cw`), and controlling the vehicle actuators within the prescribed limits ($\tau \in [-1, 1]$ and $\delta \in [-0.5236, 0.5236]$) in an open-loop control setting. The script was executed for a span of 90 seconds, while all the vehicle (and traffic light in certain cases) data were recorded with a target frequency of 30 Hz.

The eight maneuvers were performed partially automatically (τ was set constant) and partially manually (δ was controlled using human-in-the-loop teleoperation framework offered by AutoDRIVE Simulator). The input method used for steering was a standard computer mouse, which allowed continuous variation of steering proportionate to the click-and-drag distance across screen.

5.3. Dataset Structure

The recorded dataset contains an exhaustive set of features, which can be used not only for this project but also for other applications such as perception, state estimation, vehicle dynamics, motion planning, control, etc. We intentionally recorded realistically plausible sensor data from the simulator (i.e. data that can be sensed/estimated in real world as well; no unfair advantage of using simulation) to ensure the sim2real capability of our approach going forward. Following is a header describing the dataset structure:

DATA	timestamp	throttle	steering	leftTicks	rightTicks	posX	posY	posZ	roll	pitch	yaw	speed	angX	angY	angZ	accX	accY	accZ	cam0	cam1
UNIT	yyyy_MM_dd_HH_mm_ss_fff	norm%	rad	count	count	m	m	m	rad	rad	rad	m/s	rad/s	rad/s	rad/s	m/s^2	m/s^2	m/s^2	img_path	img_path

Figure 10. *Vehicle dataset structure.*

Apart from the vehicle data, the eight maneuver carried out in the Driving School scenario also comprises traffic light data:

DATA	timestamp	state
UNIT	yyyy_MM_dd_HH_mm_ss_fff	Int{0,1,2,3}

Figure 11. *Traffic light dataset structure.*

For the scope of this project, we are not considering the perception data and using just a subset of collected data for training the model since the objectives of this project do not demand such information. However, this dataset can be potentially useful as redundant information or for solving different problems concerning vehicle autonomy. Consequently, the community can benefit from this open-source dataset.

5.4. Data Balancing

The process of data collection during each simulation run generated approximately 1500 samples. This process was repeated for 25 different combinations of throttle and steering. Each combination was then tested in both clockwise and counterclockwise directions, resulting in a total of 50 different test conditions. Consequently, for each maneuver, we collected a total of over 75,000 data samples.

Out of the 5 maneuvers, the data for 3 maneuvers (skidpad, fishhook, slalom) was recorded using an automated script, which controlled the vehicle for 90 seconds in each direction.

However, since the eight maneuver comprised both clockwise and counterclockwise directions, we collected the data for approximately 180 seconds. Additionally, since the driving direction is not applicable for straight maneuver we collected data for additional throttle gradations $\tau \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$ norm%.

This way, we made sure that the collected data was uniformly distributed across all the maneuvers are and not biased to any particular maneuver. **Table 1** hosts the exact number of data samples for each maneuver.

Table 1. *Data samples across different maneuvers.*

Maneuver	Straight	Skidpad	Fishhook	Slalom	Eight	Total
Data Samples	76263	78017	76294	76894	77567	385035

5.5. Data Pre-Processing and Analysis

Straight maneuver: The straight maneuver is a vehicle dynamics test that is designed to measure a vehicle's longitudinal capacity (power and friction limits) while driving in a straight line for a predetermined time. The throttle of the vehicle was set to a constant value, $\tau \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$ norm%, and while the steering angle set to 0 rad. This made the vehicle go in a straight line based on the combination of sampled values. **Figure 12** depicts the visualization of trajectory (x, y, t), state variables (x, y, θ), rate variables (v_x, ω_z) and control variables (τ, δ) for a sample run of straight maneuver:

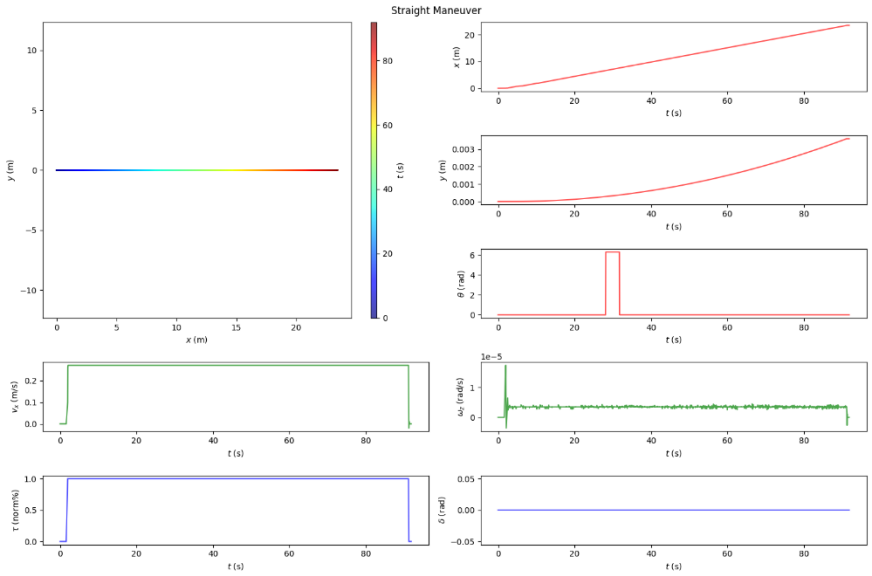


Figure 12. *Straight maneuver: data visualization for one sample run.*

Skidpad maneuver: The skidpad maneuver is vehicle dynamics test that is designed to measure a vehicle's lateral grip and handling capabilities. It involves driving a vehicle in a constant-radius circle around a circular track or pad. The throttle of the vehicle was set to a constant value, $\tau \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$ norm%, and so was the steering angle, $\delta \in \{0.1047, 0.2094, 0.3142, 0.4189, 0.5236\}$ rad. This made the vehicle go in circles of different radii $\left(R = \frac{l}{\tan(\delta)}\right)$ based on the combination of sampled values. **Figure 13** depicts the visualization of trajectory (x, y, t), state variables (x, y, θ), rate variables (v_x, ω_z) and control variables (τ, δ) for a sample run of skidpad maneuver.

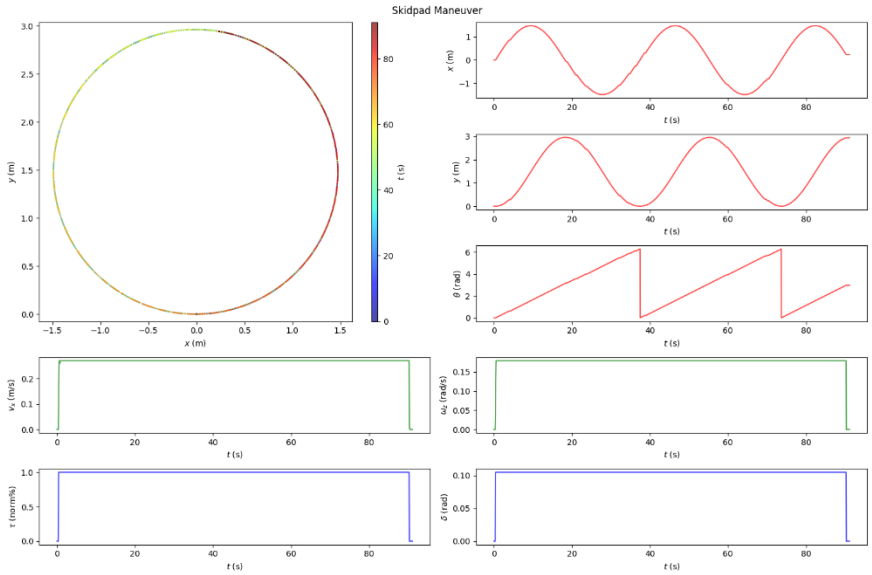


Figure 13. *Skidpad maneuver: data visualization for one sample run.*

Fishhook maneuver: The fishhook maneuver has a steering input defined in terms of angle against time. The test measures the steering gradient and sensitivity by making the vehicle go in spiral trajectory in one of the two following ways: (i) keeping the velocity constant and increasing the steering angle gradually, or (ii) keeping the steering angle constant and increasing the velocity gradually; we adopted the former method. The throttle of the vehicle was set to a constant value, $\tau \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$ norm%, while the steering angle was continuously incremented with a ramp input, $\delta = \min\left(k_{fishhook} * \text{ramp}(t), \delta_{max}\right)$ rad based on the time (t) elapsed since initialization (90 s). This made the vehicle go in spirals of different radii. **Figure 14** depicts the visualization of trajectory (x, y, t), state variables (x, y, θ), rate variables (v_x, ω_z) and control variables (τ, δ) for a sample run of fishhook maneuver.

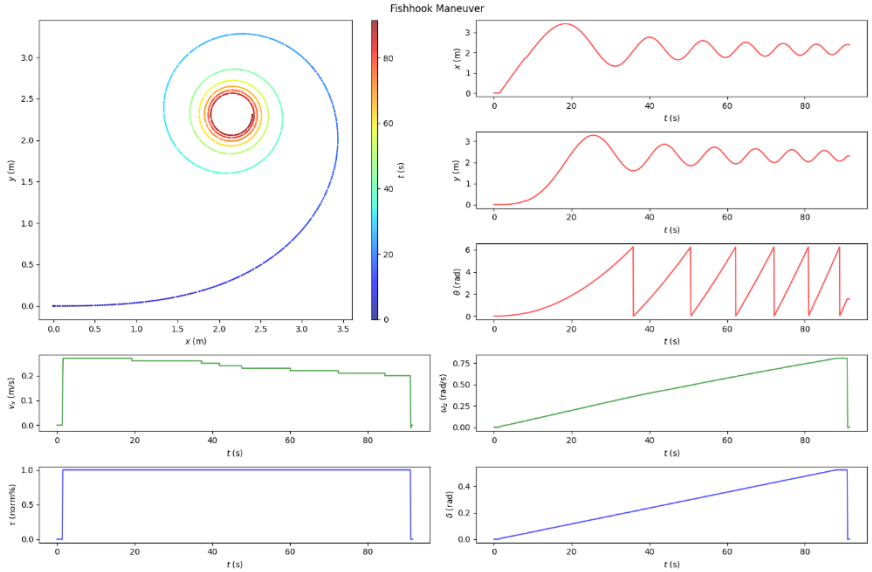


Figure 14. *Fishhook maneuver: data visualization for one sample run.*

Slalom maneuver: The slalom maneuver is a driving exercise that involves maneuvering a vehicle through a series of cones or other obstacles set up in a zigzag pattern. The objective of the slalom test is to assess a driver's ability to handle the vehicle in a tight space and maintain control while making quick turns and changes of direction. It measures the vehicle's roll and yaw stability. The throttle of the vehicle was set to a constant value, $\tau \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$ norm%, while the steering angle was controlled with a limit-scaled time-shifted sinusoidal input, $\delta = \delta_{lim} * \sin\left(\frac{\pi}{2} + t\right)$ rad based on the time (t) elapsed since initialization (90 s) and prescribed steering actuation limit $\delta_{lim} \in \{0.1047, 0.2094, 0.3142, 0.4189, 0.5236\}$. This made the vehicle go in sinusoidal trajectories of different amplitude. **Figure 15** depicts the visualization of trajectory (x, y, t), state variables (x, y, θ), rate variables (v_x, ω_z) and control variables (τ, δ) for a sample run of slalom maneuver.

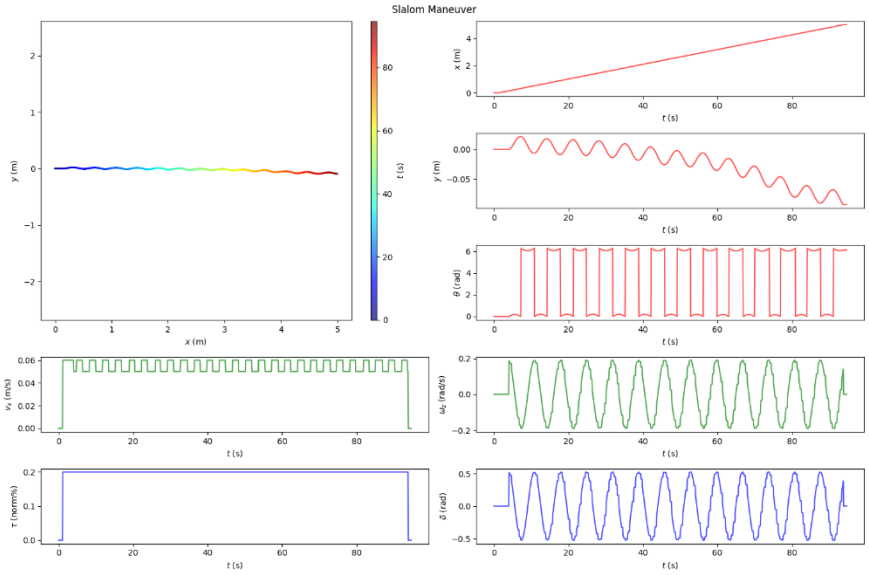


Figure 15. *Slalom maneuver: data visualization for one sample run.*

Eight maneuver: The 8-shape maneuver, also known as the figure-8 maneuver or 8-loop maneuver, is a common driving exercise used to improve a driver's handling and control of a vehicle. The maneuver consists of driving in a figure-eight pattern, which requires the driver to make continuous turns in both directions. As described earlier, the eight maneuvers were performed partially automatically (τ was set constant, $\tau \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$ norm%) and partially manually (δ was controlled using the human-in-the-loop teleoperation framework offered by AutoDRIVE Simulator). The input method used for steering was a standard computer mouse, which allowed continuous variation of steering proportionate to the click-and-drag distance across screen. **Figure 16** depicts the visualization of trajectory (x, y, t), state variables (x, y, θ), rate variables (v_x, ω_z) and control variables (τ, δ) for a sample run of eight maneuver.

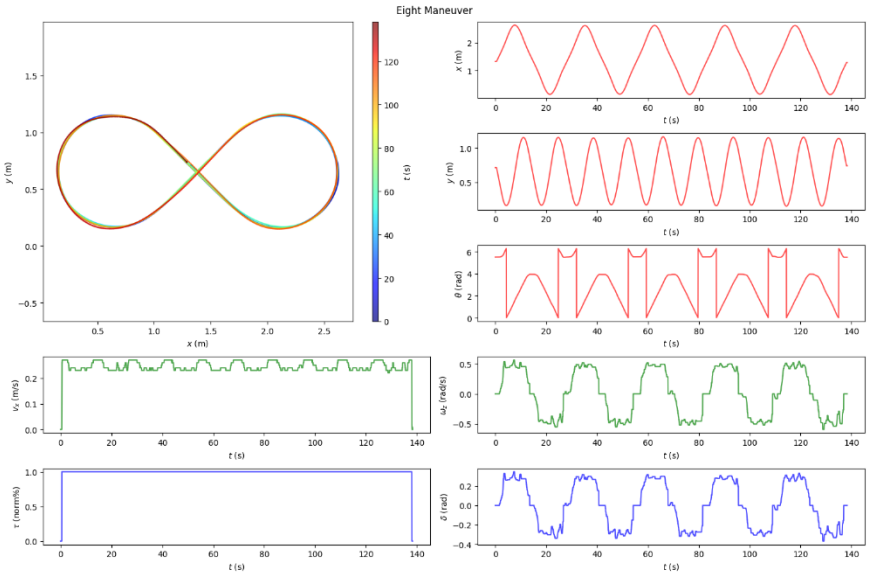


Figure 16. *Eight maneuver: data visualization for one sample run.*

5.6. Data Quality and Integrity

AutoDRIVE Simulator weighs physics and graphics simulation equally. The simulator accounts for inherent uncertainty in vehicle dynamics, sensor characteristics, actuator response as well as the vehicle-environment interaction. Externally injecting noise to the vehicle dynamics dataset is not recommended as it might adversely affect the results of the test cases collected. The AutoDRIVE simulator is designed to simulate high fidelity driving conditions and it ensures that the simulation is as close to reality as possible.

The quality of data is directly dependent on the simulation fidelity, which is evident from the observations in our dataset:

- **Straight maneuver:** a gradual increase in lateral motion is observed even when the steering input remains 0. Also, there are instances where the orientation output is 2π instead of 0.
- **Fishhook maneuver:** it is observed that the steering input had a strong correlation with the vehicle's motion in both the longitudinal and lateral directions. The change in the steering input had a significant impact on the vehicle dynamics - as the turning radius decreased, the angular velocity of the vehicle increased and the longitudinal velocity decreased.
- **Slalom maneuver:** it is observed that minor imperfections in vehicle-terrain interaction lead to variation in trajectory for the different runs of the test with similar test conditions. This emphasizes the significance of dynamics simulation as opposed to purely kinematics simulation.
- **Eight maneuver:** the vehicle is provided with a constant velocity and manual steering commands. As a result, the trajectory followed by the vehicle is similar but not exactly the same, which introduces a favorable variance in the collected dataset.

6. MODEL TRAINING

6.1. Neural Network Architecture and Hyperparameters

We experimented with various neural network architectures ranging from completely point-estimate neural networks to completely probabilistic Bayesian neural networks as well as hybrid combinations of these.

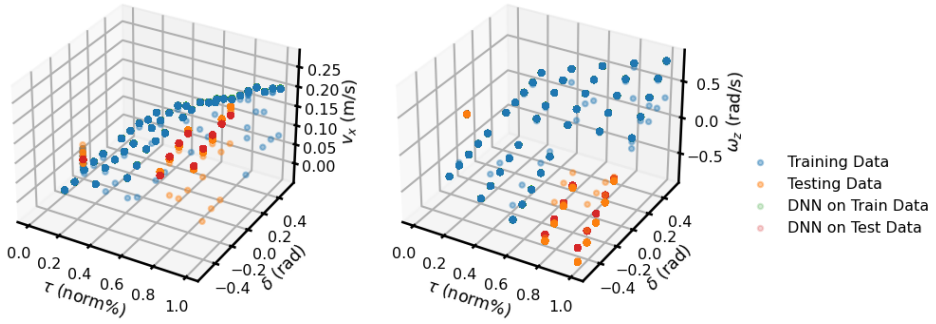


Figure 17. *Predictions on skidpad maneuver {e=5, α=1e-3, 3D, ReLU, Adam, MSE}.*

As depicted in **Figure 17**, a point estimate deep neural network of 3 fully connected layers was able to fit the skidpad data within 5 epochs using Adam optimizer and learning rate of 1e-3. However, it could not generalize well to the two operating points in the test data, which were intentionally kept hidden from it during the training process. This clearly demonstrates the inherent drawback of point estimate deep neural networks, which is that the cannot quantify the uncertainty in their predictions and can therefore be very wrong and very confident at the same time.

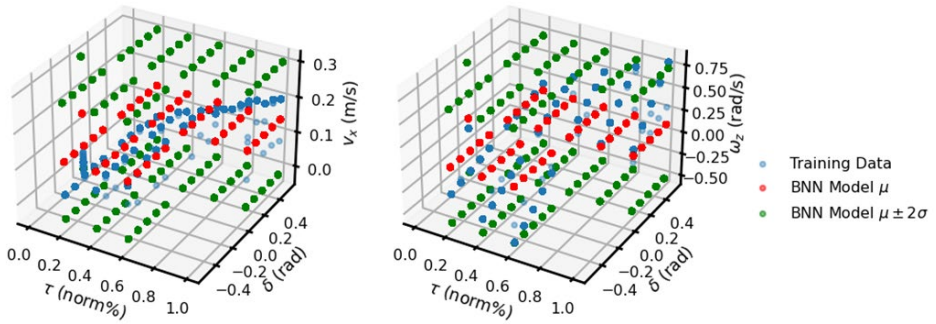


Figure 18. *Predictions on skidpad maneuver {e=10, α=1e-3, 3V, Sigmoid, Adam, NLL, MVG, approx. KLD}.*

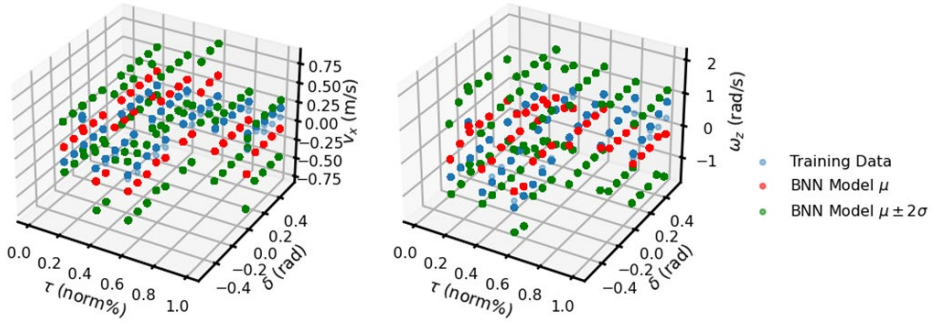


Figure 19. *Predictions on skidpad maneuver {e=25, α=5e-3, 2V, Tanh, RMSProp, NLL, MVG, approx. KLD}.*

Among the purely probabilistic Bayesian neural networks, we demonstrate the performance of a three-layer (**Figure 18**) and a two-layer (**Figure 19**) PBNN, both of which were trained for the skidpad maneuver but with different hyperparameters. It is evident that neither of the models fits the data very well. This can be explained by the inherent complexity of probabilistic Bayesian deep learning, where the goal is to learn the probability distributions of network parameters and predictions. As discussed earlier, PBNNs generally require more data and take longer to converge as compared to the standard point estimate neural networks. However, a key thing to note here is that both of these models can quantify the uncertainty in their output, which can be assessed to threshold the trust in the model predictions.

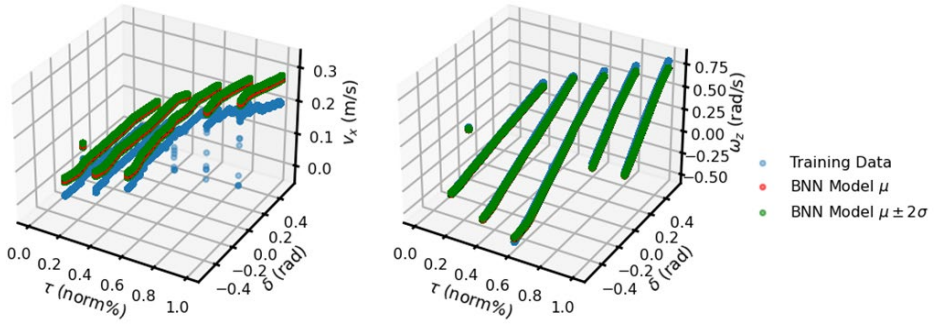


Figure 20. *Predictions on fishhook maneuver {e=25, α=1e-3, 3D-2V, Sigmoid, RMSProp, MAE, MVG, approx. KLD}.*

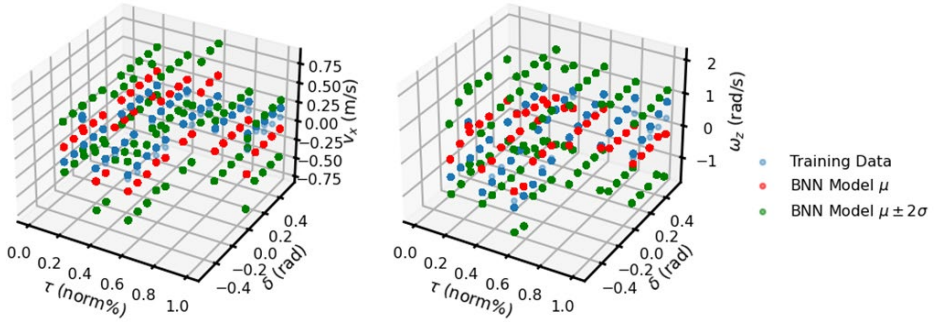


Figure 21. *Predictions on fishhook maneuver {e=25, α=1e-3, 3D-2V, Sigmoid, Adam, MAE, MVG, exact KLD}.*

Based on our observations, we could infer that point estimate deep neural networks converged very quickly whereas probabilistic Bayesian deep neural networks could quantify uncertainty in their predictions. Consequently, we tried fusing the two architectures to harvest the benefits of each (refer **Figure 20** and **Figure 21**). The hybrid architecture depicted in **Figure 22** implements a variational input layer followed by another variational hidden layer. The middle hidden layers are then standard fully connected layers followed by another variational hidden layer before the probabilistic output layer. This architecture is conceptualized to implicitly capture the uncertainty in the input, process that information effectively and also quantify the uncertainty in predictions.

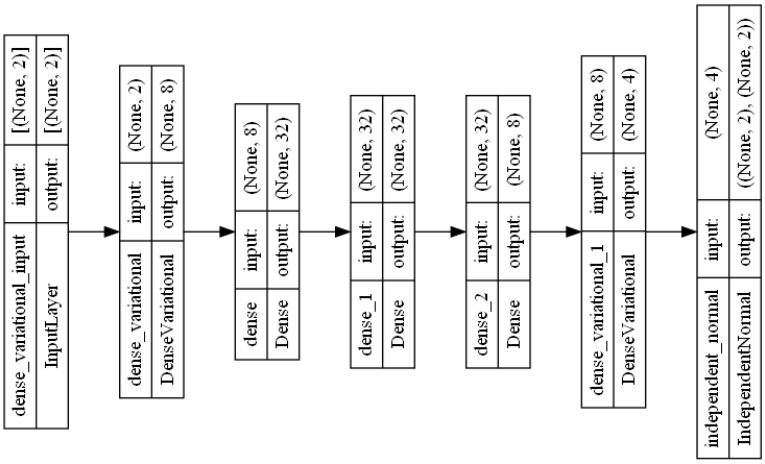


Figure 22. *Hybrid probabilistic Bayesian neural network architecture with 3 dense and 2 variational (3D-2V) layers.*

Apart from the model architecture, we also experimented with a variety of hyperparameters including different learning rates {5e-3, **1e-3**, 1e-2, 1e-1}, epochs {5, 10, 15, 20, **25**, 30}, activation functions {Tanh, **Sigmoid**, ELU, **ReLU**, SeLU, Softplus}, optimizers (SGD, **RMSProp**, Adam), loss functions {NLL, **MSE**, MAE}, prior and posterior distributions {univariate or **multivariate Gaussian**} as well as methods of computing the KL divergence {**exact**, approx.}, to name a few. The bold values in each of the aforementioned hyperparameter sets highlights the ones that showed promising results. It is interesting to see that the "go to" hyperparameters in case of standard deep learning regression problems may not be the optimal choice for probabilistic Bayesian deep neural networks.

6.2. Training Results

The model architecture and hyperparameters discussed earlier could generalize across all the vehicle dynamics maneuvers, both individually as well as collectively. **Figure 23** depicts the trends of training and testing loss metrics over the training epochs for each maneuver.

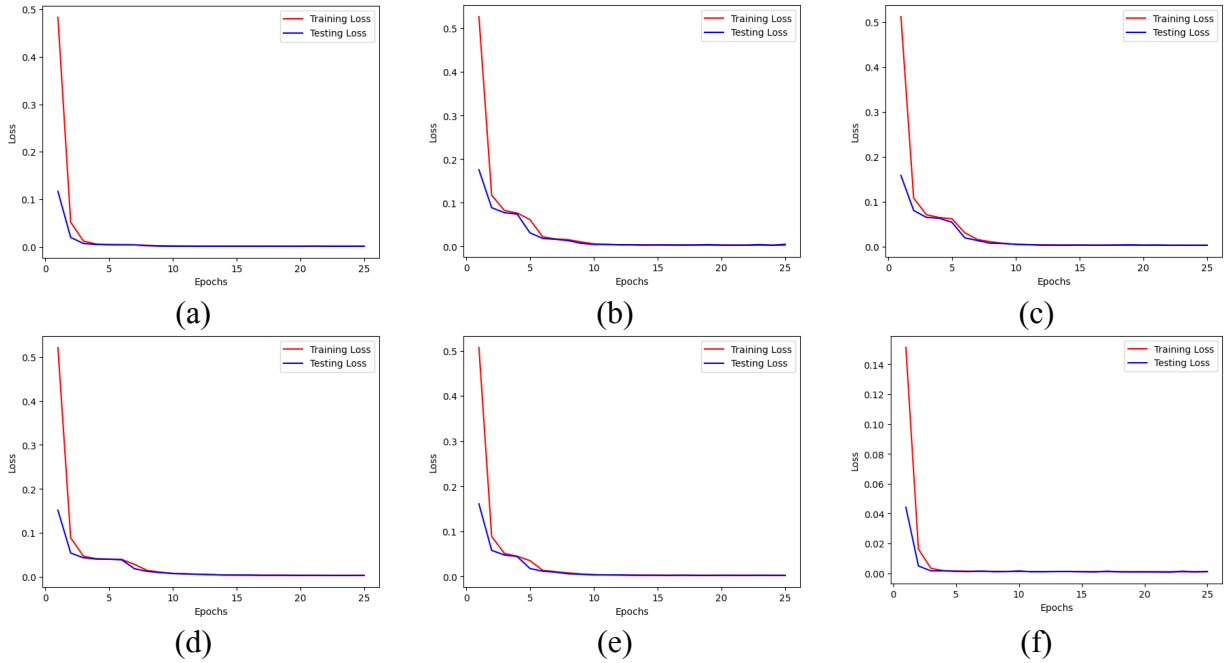


Figure 23. *Training and testing loss curves for: (a) straight maneuver; (b) skidpad maneuver; (c) fishhook maneuver; (d) slalom maneuver; (e) eight maneuver; and (f) all maneuvers.*

7. MODEL INFERENCE

7.1. Straight Maneuver

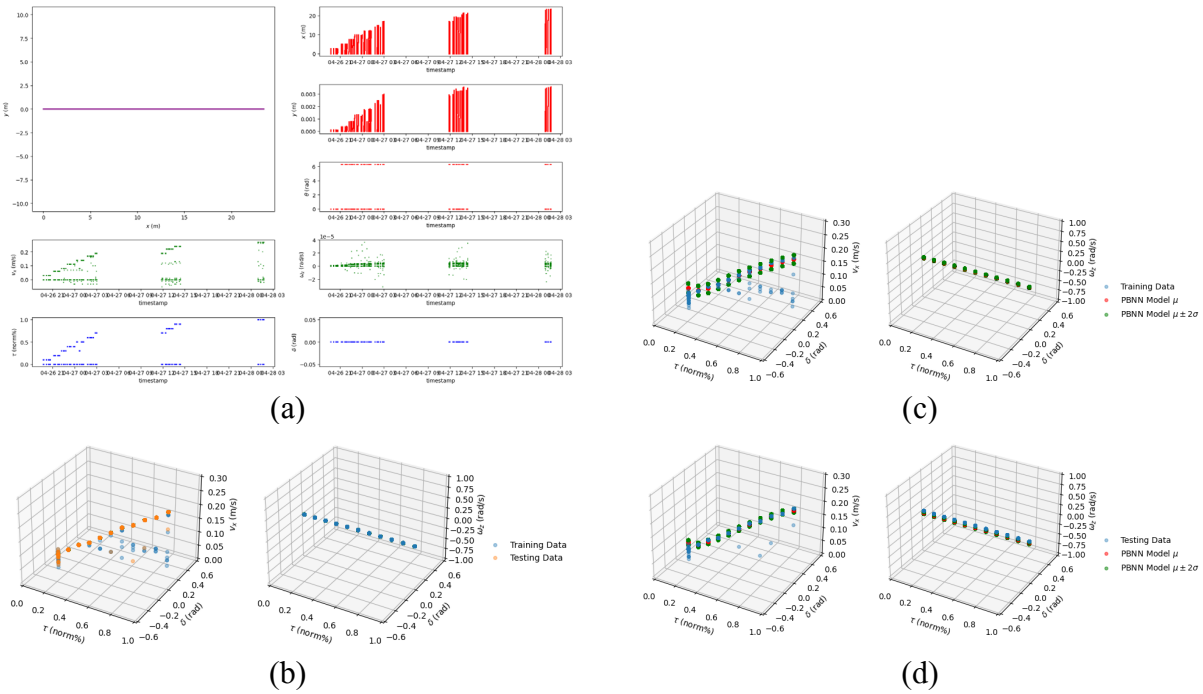


Figure 24. *Model inference for straight maneuver: (a) collective data visualization; (b) training and testing datasets; (c) PBNN performance on training data; and (d) PBNN performance on testing data.*

Figure 24(a) depicts the data distribution across various runs of the straight maneuver. The intermittent gaps between certain data samples indicate the difference in timestamp (ranging from a few minutes to a few days) of when the data runs were recorded. In addition to the spatial information, one can observe the distribution of state, rate and control variables for this particular maneuver.

Figure 24(b) depicts the training and testing datasets for straight maneuver, where the labels are independently represented w.r.t. both the features. This clearly denotes the coupled dynamics of a vehicle: the linear and angular velocities are both affected by the throttle as well as steering control inputs.

Finally, **Figure 24(c)** and **Figure 24(d)** show the model predictions on training as well as testing datasets of straight maneuver. It can be observed that the model is able to fit the data well with low uncertainty for linear velocity predictions and very low uncertainty for angular velocity predictions. This can be explained by the outliers in the linear velocity data due to the vehicle achieving steady-state condition only after a few time steps (especially for higher throttle values).

7.2. Skidpad Maneuver

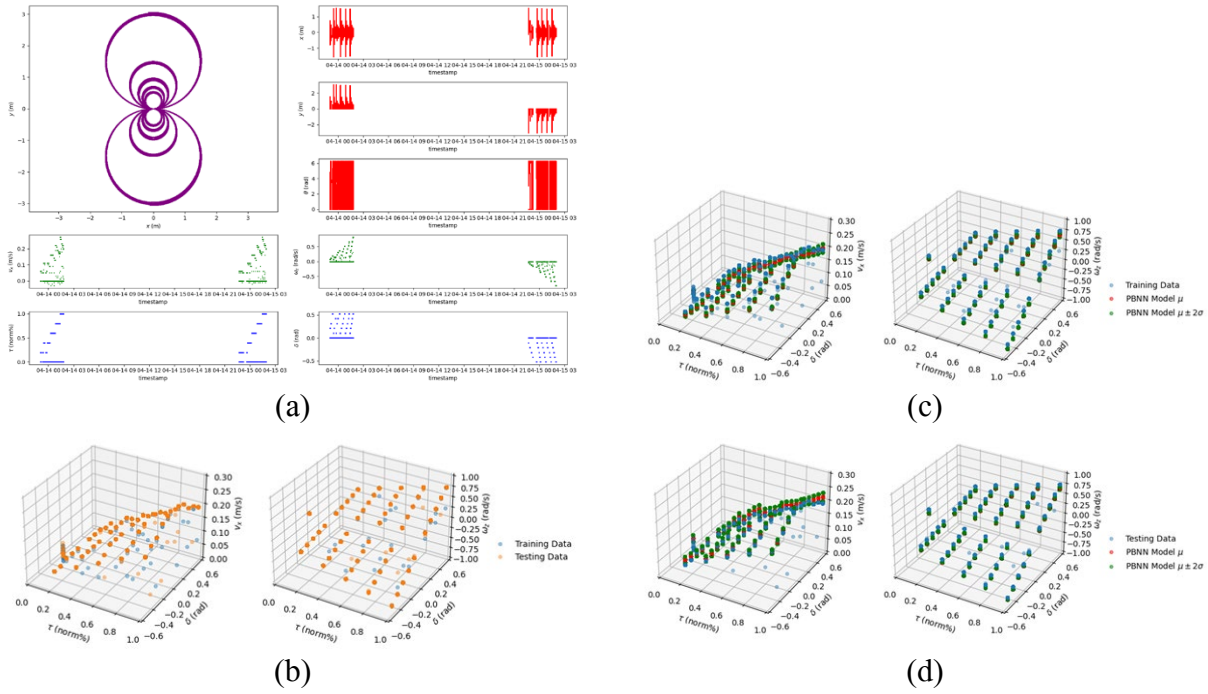


Figure 25. Model inference for skidpad maneuver: (a) collective data visualization; (b) training and testing datasets; (c) PBNN performance on training data; and (d) PBNN performance on testing data.

Figure 25(a) depicts the data distribution across various runs of the skidpad maneuver. The intermittent gaps between certain data samples indicate the difference in timestamp (ranging from a few minutes to a few days) of when the data runs were recorded. In addition to the spatial information, one can observe the distribution of state, rate and control variables for this particular maneuver.

Figure 25(b) depicts the training and testing datasets for skidpad maneuver, where the labels are independently represented w.r.t. both the features. This clearly denotes the coupled dynamics of a vehicle: the linear and angular velocities are both affected by the throttle as well as steering control inputs.

Finally, **Figure 25(c)** and **Figure 25(d)** show the model predictions on training as well as testing datasets of skidpad maneuver. It can be observed that the model is able to fit the data well with low uncertainty for linear velocity predictions and very low uncertainty for angular velocity predictions. This can be explained by the outliers in the linear velocity data due to the vehicle achieving steady-state condition only after a few time steps (especially for higher throttle values).

7.3. Fishhook Maneuver

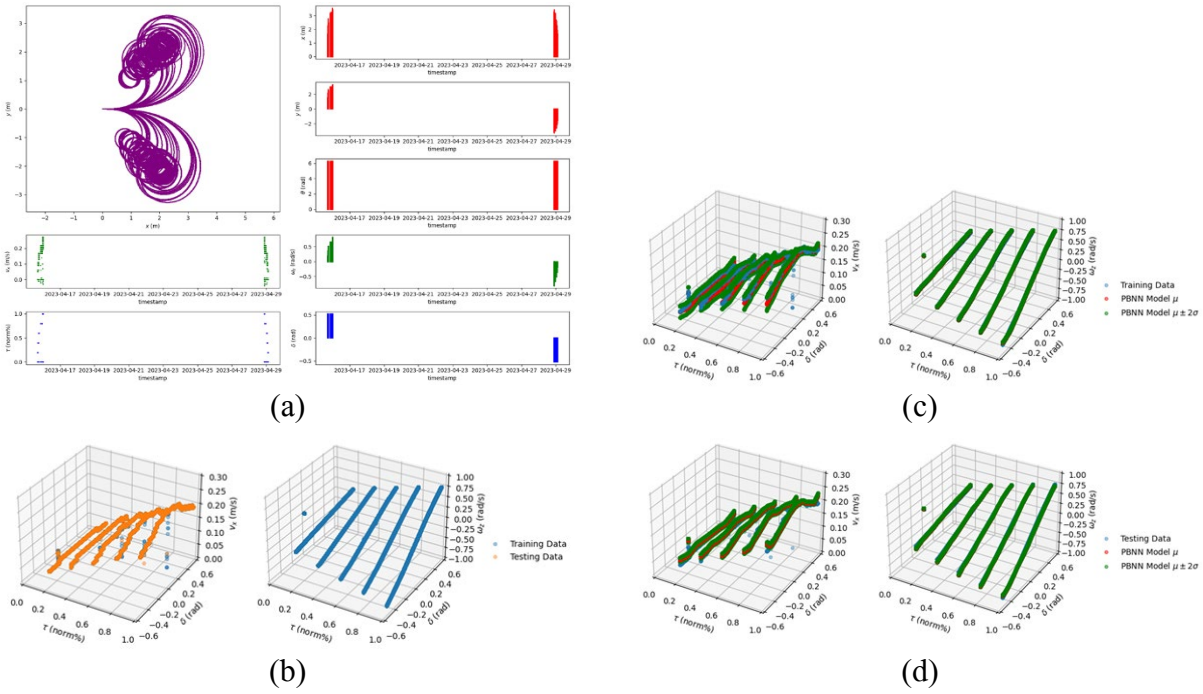


Figure 26. Model inference for fishhook maneuver: (a) collective data visualization; (b) training and testing datasets; (c) PBNN performance on training data; and (d) PBNN performance on testing data.

Figure 26(a) depicts the data distribution across various runs of the fishhook maneuver. The intermittent gaps between certain data samples indicate the difference in timestamp (ranging from a few minutes to a few days) of when the data runs were recorded. In addition to the spatial information, one can observe the distribution of state, rate and control variables for this particular maneuver.

Figure 26(b) depicts the training and testing datasets for fishhook maneuver, where the labels are independently represented w.r.t. both the features. This clearly denotes the coupled dynamics of a vehicle (more so than any other maneuver): the linear and angular velocities are both affected by the throttle as well as steering control inputs.

Finally, **Figure 26(c)** and **Figure 26(d)** show the model predictions on training as well as testing datasets of fishhook maneuver. It can be observed that the model is able to fit the data well with low uncertainty for linear velocity predictions and extremely low uncertainty for angular velocity predictions. The former can be explained by the outliers in the linear velocity data due to the vehicle achieving steady-state condition only after a few time steps (especially for higher throttle values), and the latter can be explained by the smooth variation in steering commands, which excited the lateral vehicle dynamics continuously. Another aspect to note here is that the model seems to be more certain in its predictions on the testing dataset than it is on the training dataset. This could be explained either by the data distribution while splitting the data or averaging out multiple predictions on the training dataset may possibly reduce the uncertainty of the model.

7.4. Slalom Maneuver

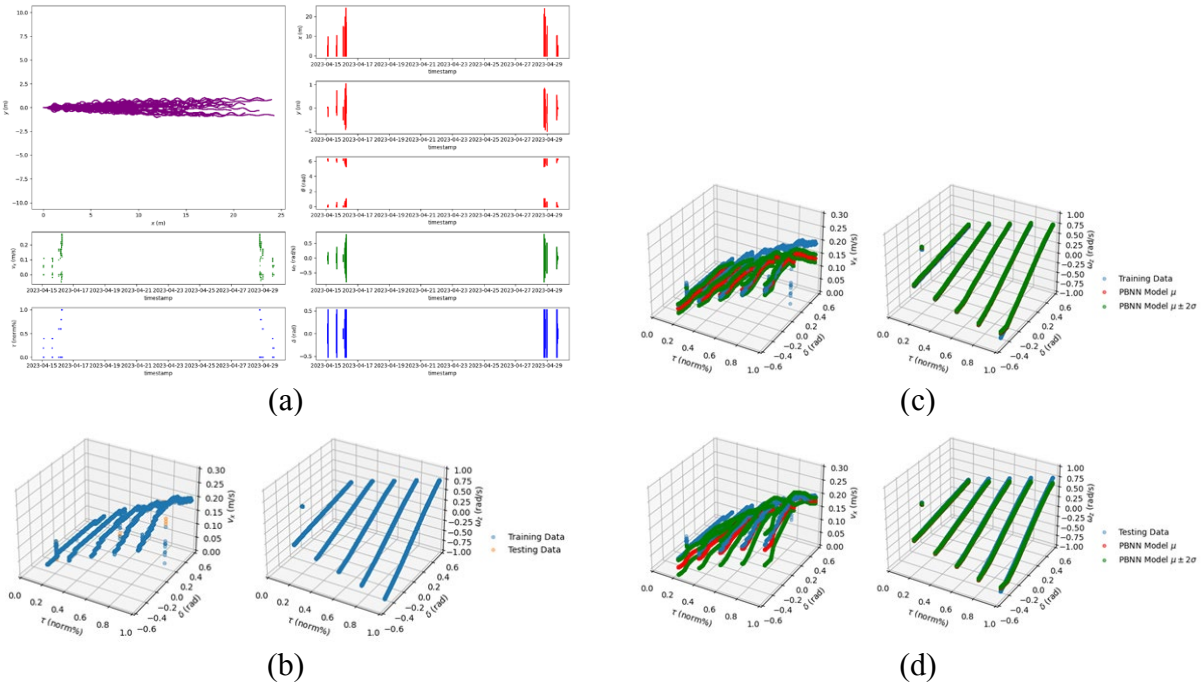


Figure 27. Model inference for slalom maneuver: (a) collective data visualization; (b) training and testing datasets; (c) PBNN performance on training data; and (d) PBNN performance on testing data.

Figure 27(a) depicts the data distribution across various runs of the slalom maneuver. The intermittent gaps between certain data samples indicate the difference in timestamp (ranging from a few minutes to a few days) of when the data runs were recorded. In addition to the spatial information, one can observe the distribution of state, rate and control variables for this particular maneuver.

Figure 27(b) depicts the training and testing datasets for slalom maneuver, where the labels are independently represented w.r.t. both the features. This clearly denotes the coupled dynamics of a vehicle: the linear and angular velocities are both affected by the throttle as well as steering control inputs.

Finally, **Figure 27(c)** and **Figure 27(d)** show the model predictions on training as well as testing datasets of slalom maneuver, perhaps the most difficult maneuver to master. It can be observed that the model is able to fit the data well with moderate uncertainty for linear velocity predictions and very low uncertainty for angular velocity predictions. This can be explained by the maneuver's nature of exiting lateral dynamics more than the longitudinal along with the fact that the linear velocity data has outliers due to the vehicle achieving steady-state condition only after a few time steps (especially for higher throttle values).

7.5. Eight Maneuver

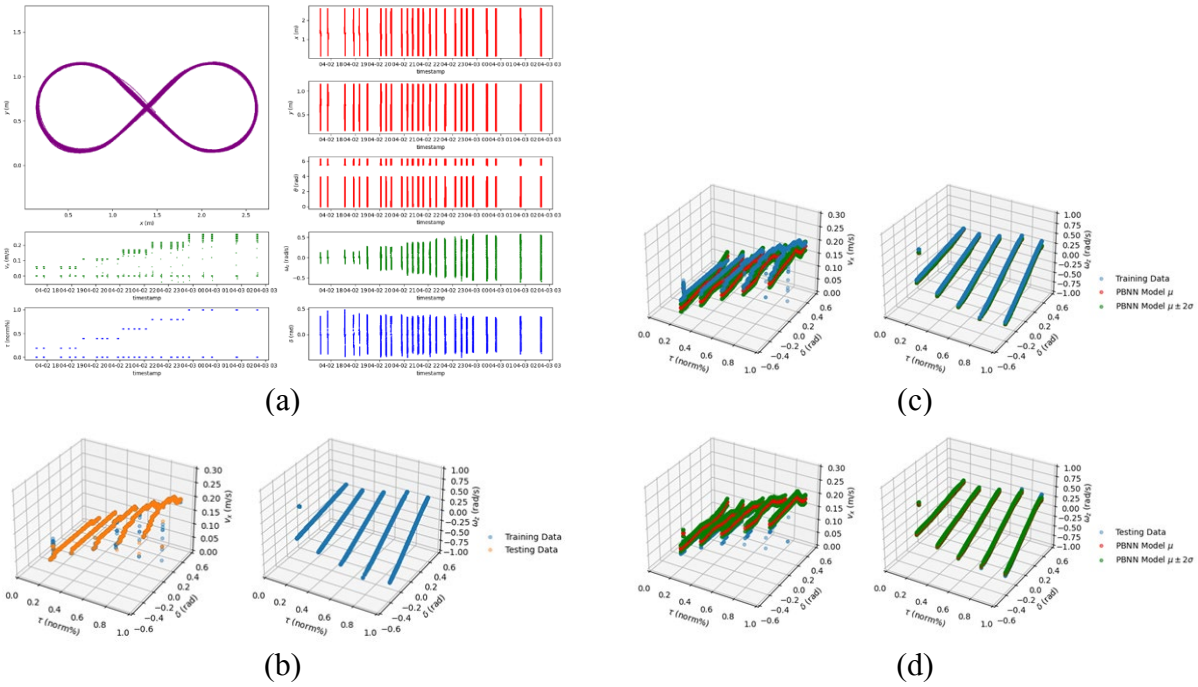


Figure 28. Model inference for eight maneuver: (a) collective data visualization; (b) training and testing datasets; (c) PBNN performance on training data; and (d) PBNN performance on testing data.

Figure 28(a) depicts the data distribution across various runs of the eight maneuver. The variance in spatial coordinates and steering input can be clearly observed, which is a result of manually driving the vehicle. In addition to the spatial information, one can also observe the distribution of state, rate and control variables for this particular maneuver.

Figure 28(b) depicts the training and testing datasets for eight maneuver, where the labels are independently represented w.r.t. both the features. This clearly denotes the coupled dynamics of a vehicle: the linear and angular velocities are both affected by the throttle as well as steering control inputs.

Finally, **Figure 28(c)** and **Figure 28(d)** show the model predictions on training as well as testing datasets of eight maneuver. It can be observed that the model is able to fit the data well with low uncertainty for linear velocity predictions and very low uncertainty for angular velocity predictions. This can be explained by the outliers in the linear velocity data due to the vehicle achieving steady-state condition only after a few time steps (especially for higher throttle values).

7.6. All Maneuvers

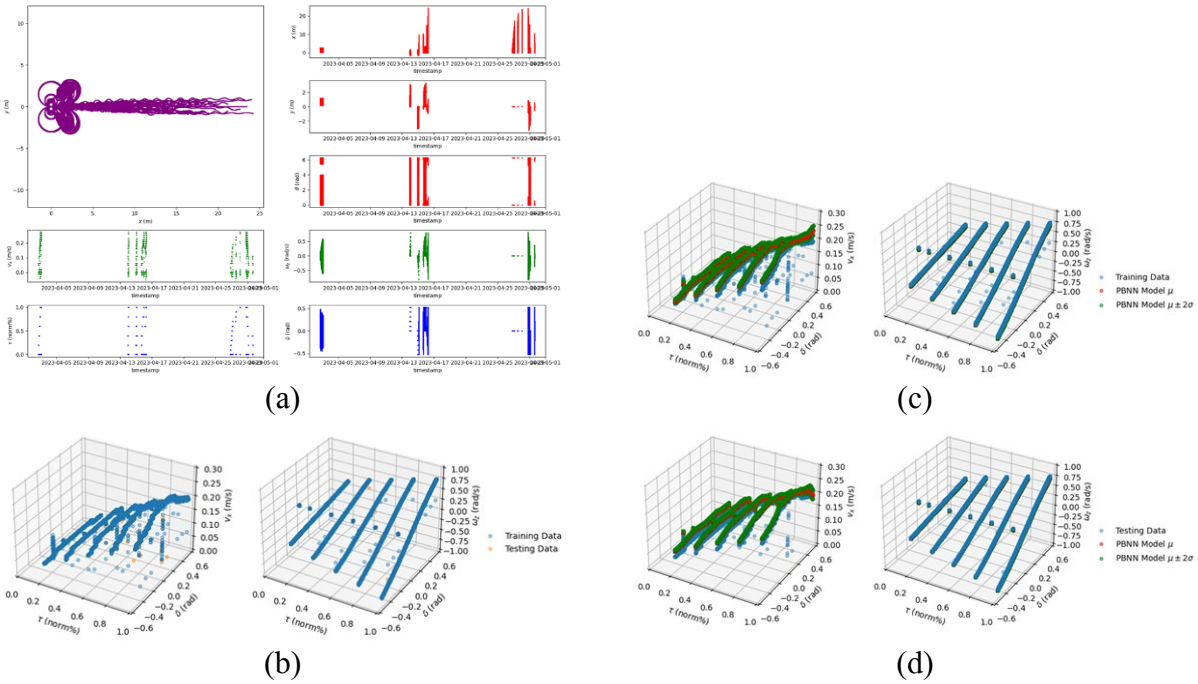


Figure 29. *Model inference for eight maneuver: (a) collective data visualization; (b) training and testing datasets; (c) PBNN performance on training data; and (d) PBNN performance on testing data.*

Figure 29(a) depicts the data distribution across various runs of all the maneuvers. The intermittent gaps between certain data samples indicate the difference in timestamp (ranging from a few minutes to a few days) of when the data runs were recorded. In addition to the spatial information, one can observe the distribution of state, rate and control variables, which span across the operational design domain of the vehicle.

Figure 29(b) depicts the training and testing datasets for all the maneuvers combined, where the labels are independently represented w.r.t. both the features. This clearly denotes the coupled dynamics of a vehicle: the linear and angular velocities are both affected by the throttle as well as steering control inputs.

Finally, **Figure 29(c)** and **Figure 29(d)** show the model predictions on training as well as testing datasets of all the maneuvers combined. It can be observed that the model is able to fit the data well with low uncertainty for linear velocity predictions and extremely low uncertainty for angular velocity predictions. This can be explained by the outliers in the linear velocity data due to the vehicle achieving steady-state condition only after a few time steps (especially for higher throttle values). It is interesting to see that the model can generalize well across all the maneuvers without underfitting the data or overfitting any one particular maneuver. This also indirectly validates the quality of the dataset.

8. CONCLUSION AND FUTURE WORK

This work investigated the application of probabilistic Bayesian deep learning methods to help bridge the gap between formal and data-driven approaches to develop computationally light-weight surrogate vehicle dynamics models that are generalizable, interpretable, and explainable. Furthermore, this work proposed a novel hybrid model architecture to leverage the combined benefits of both point estimate and probabilistic Bayesian neural networks. Particularly, the proposed approach was applied to "learn" a reduced-order (i.e., surrogate) dynamics model of an Ackermann-steered vehicle under Markov assumption given its low-level control inputs such as throttle, brake and steering, as well as rate of change of state variables. It was comparatively shown that this approach not only helps fit and generalize over the data better, but also in terms of quantification of the aleatoric and epistemic uncertainties with its predictions, thereby imbibing explainability into these models.

Some of the directions worth pursuing as an extension of this project going into the future include:

- Formulate and implement probabilistic Bayesian LSTM (with delayed embedding).
- Try different input representation techniques (e.g. timeseries snapshot of state information).
- Utilize other state (and perception) information from the current dataset.
- Transfer learning for expanding model's operational design domain (ODD) to different operating points.
- Ensemble models for different operating points.
- Simulation + real-world vehicle data (across scales – small, mid and full scale vehicles).
- Active sampling to collect (sample) new data points smartly based on epistemic uncertainty.

9. SUPPLEMENTAL MATERIALS

All source files including codebase, project proposal, presentation slides and technical reports used for the completion of various aspects of the capstone project are openly available on GitHub. Following is a link to our repository: https://github.com/Tinker-Twins/Deep-Learning-Applications-in-Engineering/tree/main/Final-Project. We have isolated notebooks for various experiments in order to avoid overloading the computational resources with a lot of data and processing operations.

Furthermore, the entire dataset (including both dynamics as well as perception data) corresponding to all variations across all maneuvers has been publicly released and is openly available on GitHub. Following is a link to our repository: https://github.com/Tinker-Twins/AutoDRIVE-Nigel-Dataset. It is worth mentioning that this repository is about 66 GB in size and although it does NOT require Git LFS, the users are kindly advised to check their internet data plan and local disk space before cloning this repository.

Although the primary aim of using a Git repositories for this project was to enable version control and collaborative development, this also gave us the opportunity to document all the progress using the markdown files (README.md). Despite challenges and scarcity of time, we invested a significant amount of time and effort in building and maintaining our repository throughout the duration of this project. We hope that this repository will help the professor and the teaching assistant (TA) in better understanding our efforts, outcomes and learnings from this project and potentially serve as a documentation for us (or others) when trying to explore similar approaches. That being said, we completely respect the academic integrity moral code and as such in no way promote direct use of any part of our repository (as a whole or in part) by current and/or future students taking this course.