

Aalto University  
School of Science  
Master's Programme in ICT Innovation

Rui Teng

# Deep Learning for Error Modeling of Tractor-semitrailer Dynamics Model

Master's Thesis  
Espoo, September 8, 2022

Supervisor: Assistant Professor Arno Solin, Aalto University  
Advisor: Mohamed Takkoush, M.Sc., Volvo Autonomous Solutions  
Hans-Martin Heyn, Ph.D., University of Gothenburg  
Thorsten Helfrich, Ph.D., Volvo Autonomous Solutions

<b>Author:</b>	Rui Teng	
<b>Title:</b>	Deep Learning for Error Modeling of Tractor-semitrailer Dynamics Model	
<b>Date:</b>	September 8, 2022	<b>Pages:</b> 51
<b>Major:</b>	Data Science	<b>Code:</b> SCI3095
<b>Supervisor:</b>	Assistant Professor Arno Solin	
<b>Advisor:</b>	Mohamed Takkoush M.Sc. Hans-Martin Heyn, Ph.D. Thorsten Helfrich, Ph.D.	
<p>In the field of autonomous driving, vehicle models are the basis for a variety of research. Vehicle models provide simulated data describing trajectories and dynamics of vehicles, and the property that the utility of vehicle models is independent of real vehicles makes it possible to obtain a large amount of data quickly. Efforts have been made surrounding the construction and validation of vehicle models. However, currently, vehicle models could produce simulation errors under certain circumstances such as extreme speed driving and complex steering.</p> <p>The proposed solution to the simulation error problem in this thesis is to predict the error and combine vehicle model simulation and error prediction. In this paper, attempts that applying deep learning to build point prediction error models and uncertainty-aware error models are made. Valid error models that pass validation tests are expected to offset residual between vehicle models and real systems, which allows using of non-accurate vehicle models. Besides, error prediction is useful to collect cases that have high simulation errors, which are important to analyze and fix vehicle models.</p> <p>The above methods are applied to a tractor-semitrailer dynamics model based on physical principles, provided by Volvo Autonomous Solutions. Involved data is collected from the corresponding tractor-semitrailer, under various driving contexts, and over diverse manoeuvres. Statistics-based evaluation results show that deep learning is potential in vehicle model error modeling, although the uncertainty-aware error model trained for the tractor-semitrailer fails in the validation test.</p>		
<b>Keywords:</b>	error modeling, deep learning, recurrent neural network, uncertainty-aware model, error reduction, tractor-semitrailer dynamics models, model validation	
<b>Language:</b>	English	

# Acknowledgements

I would like to express my deepest appreciation to my parents. Thousands of miles away in China, they work diligently and support me unreservedly.

I would also like to thank my advisor, Mohamed Takkoush. Thank you for providing me with the knowledge of vehicles. Your responsible attitude and passion for your job made me rethink the meaning of work to myself. Your enthusiasm for work subtly encouraged me to pursue a career I love. Also, thank you for cheering me up when I was down.

I also wish to thank Thorsten Helfrich, Hans-Martin Heyn, and Arno Solin for your advice on the thesis. The completion of the thesis is inseparable from your guidance. Words cannot express my appreciation for you.

I appreciate the opportunity to work on this thesis, which is the end of my past life. I believe that I will have the courage to face ordinary life in the future.

Espoo, September 8, 2022

Rui Teng

# Abbreviations and Acronyms

ANN	Artificial Neural Network
BNN	Bayesian Neural Network
BP	Back-propagation
CR	Coverage Rate
DNN	Deep Neural Network
DL	Deep Learning
HMC	Hamiltonian Monte Carlo
LR	Learning Rate
LSTM	Long Short-term Memory
MC	Monte Carlo
MSE	Mean Square Error
NLL	Negative Log-likelihood
OOD	Out of Distribution
RNN	Recurrent Neural Network
V&V	Verification and Validation

# Contents

<b>Acknowledgements</b>	<b>3</b>
<b>Abbreviations and Acronyms</b>	<b>4</b>
<b>1 Introduction</b>	<b>7</b>
1.1 Problem Statement . . . . .	7
1.2 Structure of Thesis . . . . .	8
<b>2 Background</b>	<b>9</b>
2.1 Model Validation . . . . .	9
2.2 Model Error Modeling . . . . .	10
2.3 Recurrent Neural Network . . . . .	11
2.4 Uncertainty-aware Deep Neural Network . . . . .	13
<b>3 Problem Formulation</b>	<b>17</b>
3.1 Volvo Dynamics Vehicle Model . . . . .	17
3.2 Vehicle Model Residual . . . . .	18
3.3 Error Model . . . . .	20
3.4 Error Model Evaluation and Validation . . . . .	21
<b>4 Materials</b>	<b>22</b>
4.1 Tractor-semitrailer Dynamics Model . . . . .	22
4.2 Data Collection . . . . .	23
4.3 Sensor Data Pre-processing . . . . .	25
4.4 Vehicle Model Simulation . . . . .	29
<b>5 Methods</b>	<b>30</b>
5.1 Error Model Architecture . . . . .	30
5.2 Training Inputs and Targets . . . . .	31
5.3 Loss Functions and Evaluation Metrics . . . . .	32
5.4 Hyper-parameter Configuration . . . . .	34

5.5	Validation Criteria . . . . .	35
<b>6</b>	<b>Results</b>	<b>36</b>
<b>7</b>	<b>Discussion</b>	<b>40</b>
7.1	Explanation of Experiment Results . . . . .	40
7.2	Shortcomings and Future Plans . . . . .	41
7.3	Potential Application of Error Models . . . . .	42
<b>8</b>	<b>Conclusions</b>	<b>43</b>
	<b>References</b>	<b>43</b>
<b>A</b>	<b>First appendix</b>	<b>47</b>
A.1	Uncertainty-aware Algorithms . . . . .	47

# Chapter 1

## Introduction

Vehicle models are the cornerstone in the field of autonomous driving to some extent, as multiple autonomous driving research tasks rely on the vast amount of simulated data provided by vehicle models. Ideal vehicle models are expected to have great performance in imitating a real system, which is represented by low simulation error. However, error is inevitable within the process of simulation. This thesis revolves around error prediction, which is useful for multiple applications including data collection, model analysis, etc. This chapter introduces the motivation and research questions of the project.

### 1.1 Problem Statement

During the simulation process of a tractor-semitrailer dynamics model created by Volvo Autonomous Solutions, whose essence is to imitate the states of real tractor-semitrailers and represent states with dynamic signals for given manoeuvres in specified environments, error is inevitable to be produced, especially under extreme circumstances such as high speed driving. The existence of error could result in failing in validation tests, which signifies rejection of the vehicle model. Thus, the main purpose of this thesis is to build error models, which learn and predict the residual between the tractor-semitrailer dynamics model and the real system. The motivation is that, firstly, valid error models are able to complement the vehicle models [12] by offsetting the residual, thus, more reliable simulation of the vehicle is accessible. Secondly, error prediction allows for efficiently collecting cases with high simulation error, which are useful to analyze and fix vehicle models. Besides, error modeling is an implicit and intuitive way to validate vehicle models [12]. In a word, error modeling is worth studies.

As vehicle states are dynamic, temporal-based regression methods are attempted in this project. Based on the above-mentioned vehicle model and data collected

from the corresponding vehicle, recurrent neural network (RNN) based point prediction models are built to predict model error. Besides, to have insights into the uncertainty level about error prediction, RNN based uncertainty-aware models that provide predictive interval are also established.

To prove that the error models have a good performance in predicting errors, evaluation methods based on statistic analysis are implemented on test data. Besides, validation test id designed for uncertainty-aware models to check if they are trustworthy for complementing the vehicle model,

In summary, the research goal is

1. to explore the possibility of applying deep learning (DL) for predicting vehicle model error for the tractor-semitrailer dynamics models.
2. to investigate if DL-based error models are accepted to complement vehicle models.

## 1.2 Structure of Thesis

Chapter 2 introduces background knowledge related to the project purpose, including aspects of neural networks, model validation, and error modeling. Chapter 3 elaborates the project purpose in detail and builds a connection between background knowledge and this project. Chapter 4 shows the materials including the vehicle model and data involved in this project. Chapter 5 introduces experiment methods, including model architecture, parameters selection, etc. Chapter 6 presents the evaluation and validation results of the error models. Discussion around results and potential future research direction is in chapter 7. Finally, chapter 8 concludes the project.

## Chapter 2

# Background

To validate the vehicle model as well as error models, various validation methods are introduced in Section 2.1. For vehicle models that fail in validation tests, error models as introduced in Section 2.2 play a role in complementing vehicle model. Besides, papers regarding recurrent neural networks and uncertainty-aware neural networks, which are core algorithm of error modeling, are presented in Section 2.3 and 2.4.

### 2.1 Model Validation

Model validation is the process to determine if the model should be accepted or rejected. Following papers define various validation methods, including referring to statistic metric [13, 17], Bayesian hypothesis testing and [3] reachset conformance [25].

In [21], the Verification and validation (V&V) framework was first proposed. It gave a definition of model validation, which is "to substantiate that a computerized model within its domain of applicability possesses a satisfactory range of accuracy consistent with the intended application of the model". This means model validation is meaningful only if it is conducted within the domain of desired application of the model. The history review of V&V is provided in [20], which summarises the research regarding V&V since the 1970s.

Multiple model validation techniques were summarised in [11]: face validation, tracing, internal validation, sensitivity analysis, historical validation, predictive validation, events validation, turning tests, spectral analysis, experimentation, and convergent validation. Among these techniques, historical validation, which refers to checking if the model behaves as the real system does base on data, is of most interest in this thesis.

An example of historical validation is in [17]: to validate a full vehicle model,

coast down, step steer, and ramp steer was executed on real vehicle and virtual model to extract dynamic behavior data. To quantify the validation, RMSE calculation was conducted on sequential data.

Besides RMSE, there are other measurements derived from residual analysis. Ljung and Hjalmarsson introduced following approaches to model validation based on residual analysis in [13], where they also discussed how the distance between the model and 'true' description could be estimated. Firstly they defined several statistics around model residuals including 1) maximal absolute value of the residuals, 2) mean, variance, and mean square of the residuals, and 3) correlation between residuals and past inputs. These statistical measures are "indicators" that indicate how the model will perform in the future. After choosing measure metrics, following criterion are set to validate models: 1) upper bound for mean square error or the maximal absolute value of the residuals, and 2) correlation between residuals and past input to be small in some sense. Models that meet above requirements could be seen as unfalsified.

To validate Volvo's vehicle model, Dineff proposed statistics-driven method in [3]. Bayesian hypothesis testing applied to the location parameter of distribution that describes the observed data was conducted in the form of parameter estimation to validate the model. In other words, the posterior distribution of location parameter was constructed and hypothesis testing was then applied to the highest density interval and region of practical equivalence. The hypothesis testing results show whether the specific models are valid. Focused on the same task, Takkoush proposed a validation method by reachset conformance between low and high-fidelity models in [25]. While conducting reachability analysis on low order models, rapidly-exploring random trees are implemented to explore the dynamics of higher order models. By checking whether the performance of the high order model can be predicted from this reachability analysis, validation of models derived for specific driving context is accomplished.

## 2.2 Model Error Modeling

Model error constitutes part of model residual in conjunction with disturbance, and their relation is shown in figure 3.3. It is reasonable to divide residual between simulated data and system data into model error and disturbance, while only model error is related to model inputs [13].

Ljung put forward that model error modeling is implicitly a method for model validation and it allows better visualization of residual analysis in [12]. The falsified model is defined in their paper as the models that do not pass certain validation tests that examine their performance on given data sets. Besides, the authors believe that error models could allow use of falsified models by combining error

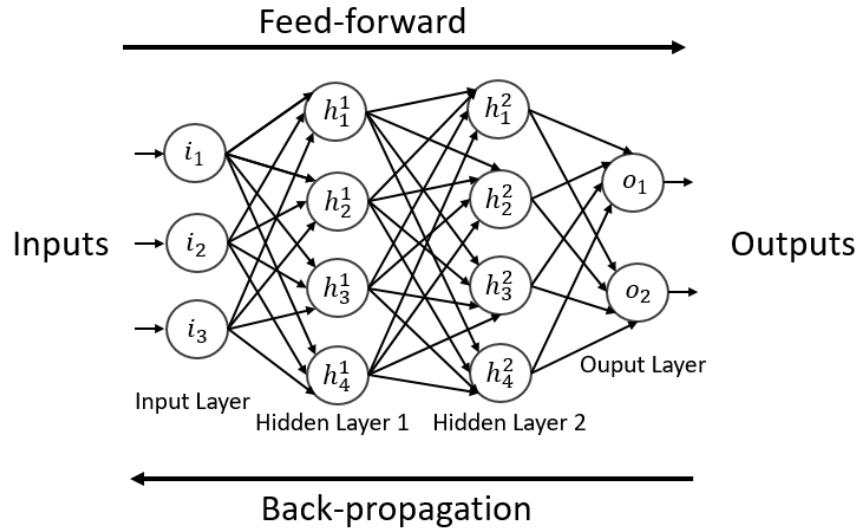


Figure 2.1: Structure of a multi-layer ANN model, which is comprised of an input layer, an output layer, and two hidden layers. Each layer contains multiple neurons that are connected by directed edges with variable weights.

models and falsified models. Linear and non-linear model error modeling solutions are presented in the paper. For dynamic models, the corresponding error models predict error using past inputs, and its linear solution could be Least-Square estimation while the non-linear solution could be neural networks. The authors suggest building linear error models first and extracting it from the residual, the rest residual could be predicted with a neural network.

## 2.3 Recurrent Neural Network

Artificial neural networks (ANN) refer to the complex network structure formed by interconnecting a large number of processing units (neurons), and it is an abstract simulation of the structure and operation mechanism of human brains. An ANN model that is comprised of multiple layers is also referred to as a Deep Neural network (DNN). Figure 2.1 presents a four-layer ANN model. As the flow shown in the figure, inputs in the form of embedding vectors first enter the model from the input layer, after operations of values on neurons and edges at each layer, outputs are provided finally.

The approximation of ANN models to data is achieved by adjusting the weights on edges, and back-propagation (BP) [18], a widely used algorithm for training neural networks, is a method for this purpose. Firstly, a loss function that describes

the distance from predicted outputs to targets for the network is defined. For instance, mean square error (MSE) as Equation 2.1, where  $y_i$  and  $\hat{y}_i$  is the  $i^{th}$  target and prediction, is an optional loss function for regression tasks.

$$MSE = \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (2.1)$$

Then, for each pair of input and target, the network calculates the loss between target and output predicted by the network, and applies gradient descent on weights, which means computing the gradient of the loss function with respect to the edge weights of the network. Iterating this process with training data set could improve model performance and reduce the loss function. BP could not only be applied to single input-output pair and iterated throughout the whole data set but also able to calculate with batches of samples.

The training goal of ANN models is to build a model that has the best prediction performance on the validation set, by figuring out the best hyper-parameter configuration over possible hyper-parameter spaces. To achieve that, evaluation is implemented on the validation set. More knowledge regarding evaluation, including avoiding overfitting, the trade-off between bias and variance, regularization, etc. is available in the book [16].

As introduced before, Recurrent Neural Network (RNN) is an option for non-linear error models. RNN [4] is a class of artificial neural networks (ANN). Similar to ANN, it takes in input vectors and gives predictions for the target. Differing from normal ANN, its connections between nodes form a graph along a temporal sequence as shown in figure 2.2. Simply put, at each time step, inputs are fed into the RNN network, and information goes through the hidden layer until the output layer, while output is produced from the output nodes, the information of the hidden layer will remain at the hidden nodes. Till the next input vector is fed in, the hidden information will be engaged in the calculation of the hidden layer. Thus, this structure allows the information to be conveyed through time. In other words, the output is decided by both the current input and hidden information got from earlier data.

Long Short-Term Memory Neural Network (LSTM) is one type of RNN model. During the training of RNN models, as the training time extends and the number of network layers increases, the problems of gradient explosion or gradient disappearance occur, which makes it impossible to process long sequence data and thus cannot obtain long-distance data information. In 1997, LSTM was proposed [9], which effectively solved above problems. In 1999, Felix et al. found that for the LSTM proposed in [9], when processing continuous input data, the state inside the network is not reset, which is possible to result in the crash of the network eventually. Therefore, they introduced a forget gate mechanism based on the ar-

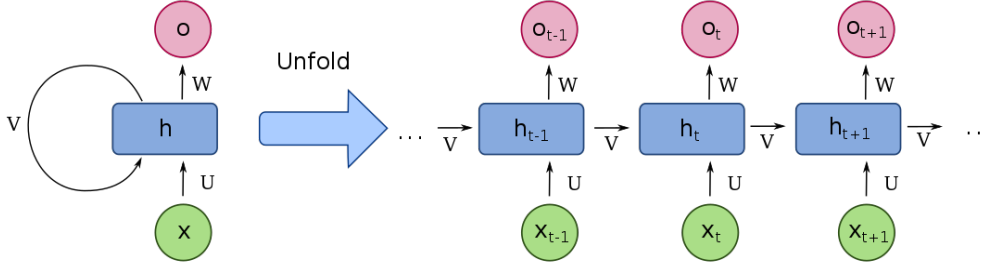


Figure 2.2: Structure of a basic RNN cell [6].  $\mathbf{x}, \mathbf{h}, \mathbf{o}$  denotes input state, hidden state, output state, respectively.  $\mathbf{U}, \mathbf{V}$  and  $\mathbf{W}$  are the weights of the network. Unfolding the structure, it shows the process of information transmitting through time. Due to the structure, previous information is accumulated inside the RNN cell and is engaged in deciding future outputs.

chitecture [9], which enables LSTM to reset its own state [8]. The structure of LSTM is presented in figure 2.3 and the corresponding computation process is shown as equation 2.2, where  $\odot$  denotes the element-wise multiplication. Compared to Basic RNN, LSTM has additional cell states that transmit through time as well as hidden states, while only hidden state is the output from the LSTM cell to the next layer in network

$$\begin{aligned}
 \mathbf{f}_t &= \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f) \\
 \mathbf{i}_t &= \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i) \\
 \mathbf{o}_t &= \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o) \\
 \hat{\mathbf{c}}_t &= \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c) \\
 \mathbf{c}_t &= \sigma(\mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \hat{\mathbf{c}}_t) \\
 \mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
 \end{aligned} \tag{2.2}$$

## 2.4 Uncertainty-aware Deep Neural Network

Neural networks are widely used for complex tasks. Due to the complexity of the research problem and the limitation of available data, there is no guarantee that the prediction of networks is correct all the time. Thus, an indicator that tells the uncertainty, or confidence in the opposite, about the prediction is necessary. In other words, for each input sample, along with the prediction result, there should be a value indicating how uncertain or confident the network is about the provided

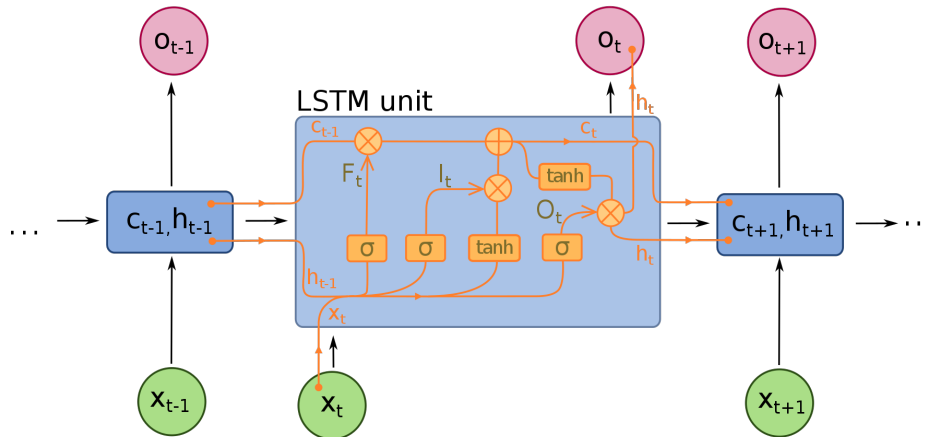


Figure 2.3: Structure of a basic LSTM cell [5]. As presented in equation 2.2, current hidden state and cell state is decided by inputs  $\mathbf{x}_t$ , previous hidden state  $\mathbf{h}_{t-1}$ , and previous cell state  $\mathbf{c}_{t-1}$ .

prediction. The following papers introduce types of uncertainty and an approach to measure uncertainty applicable to regression tasks.

## Uncertainty Type

Firstly, [2] summarised the source and categories of uncertainty for DNN predictions. There are various sources of uncertainty, including the selection of form of the model, estimation of model parameters, and inherent uncertainty of observations, etc. A simple way for uncertainty categorization is to distinguish uncertainty as either aleatory uncertainty or epistemic uncertainty.

1. Epistemic uncertainty is also referred to as model uncertainty, which could be resulted from the imperfection structure of the model, sub-optimal estimation of model parameters, and insufficient training data that cannot represent the entire context for autonomous driving tasks, such as weather, humidity, and a large number of other factors. Theoretically, epistemic uncertainty could be avoided by training enough data and getting a perfect model. However, in practice, it is impossible to include the whole context in training data. On the other hand, it is also hard to find the optimal model configuration among the huge possible parameter space.
2. Aleatoric uncertainty captures the inherent uncertainty of samples. Simply put, it is not possible to reduce aleatory uncertainty even if the networks are perfect.

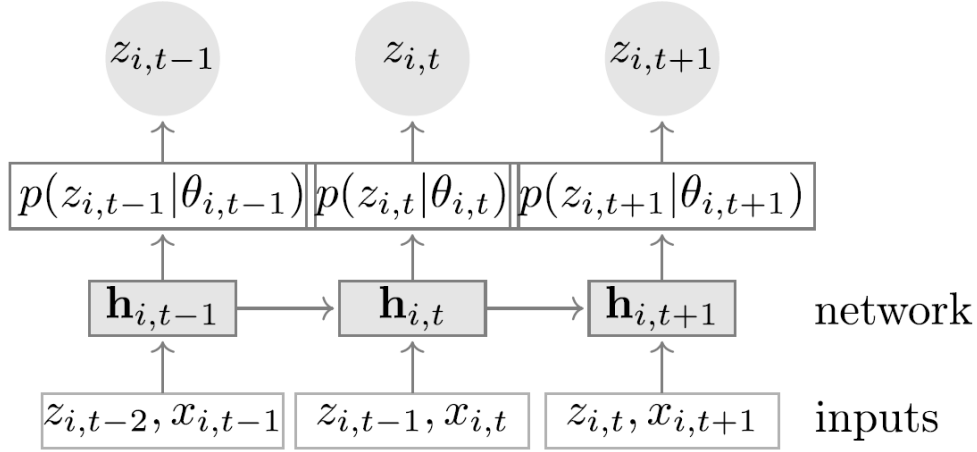


Figure 2.4: DeepAR structure. Instead of point prediction, DeepAR calculates probability distribution over possible outputs, through a probabilistic output layer, whose essence is to predict necessary parameters of pre-defined distribution type.

In summary, there are two types of uncertainty existing: aleatory uncertainty and epistemic uncertainty. To capture and quantify the uncertainty, a popular idea is to calculate a probability distribution over all possible output values and extract uncertainty from the distribution. The following paper discusses an approach that is applicable to DNN models to calculate the distribution.

## Probabilistic Layer

To measure the uncertainty for DNN prediction, one of the approaches is to introduce a probabilistic layer to the networks. Based on LSTM, Amazon proposed DeepAR, an autoregressive recurrent network that is capable to provide probabilistic forecasting [19]. Instead of point prediction, they proposed this network to predict a probability distribution of target variable given past information, thanks to the architecture which includes a probabilistic layer as shown in figure 2.4.

The type of predictive probability distribution is predefined according to the research problem, such as Gaussian distribution. In this paper, the authors took Gaussian distribution as an example as equation 2.3. The essence of the probabilistic layer is predicting distribution parameters that describe distributions, such as mean and standard deviation for Gaussian distribution. Parameters of predictive distribution could be parameterized as a linear or non-linear function of hidden information. For instance, the mean and standard deviation of Gaussian distribution in the paper is calculated as shown in equation 2.4 and 2.5.

$$p(z|\mu, \sigma) = \frac{-1}{\sqrt{2\pi\sigma^2}} \times \exp\left(\frac{-(z - \mu)^2}{2\sigma^2}\right) \quad (2.3)$$

$$\mu(\mathbf{h}_{i,t}) = \mathbf{W}_\mu^\top \mathbf{h}_{i,t} + b_\mu \quad (2.4)$$

$$\sigma(\mathbf{h}_{i,t}) = \log(1 + \exp(\mathbf{W}_\sigma^\top \mathbf{h}_{i,t} + b_\sigma)) \quad (2.5)$$

The weights  $\mathbf{W}_\mu$ ,  $\mathbf{W}_\sigma$ ,  $b_\mu$ , and  $b_\sigma$  in above equations are trained during the training process of networks, along with other weights of the network. For this purpose, the optional loss function is negative log-likelihood of target given predicted probability distribution as equation 2.6, which takes into account not only the accuracy of the point prediction but also how descriptive the associated uncertainty (variance) is. In other words, the goal is to maximize the probability of occurrence of the actual target given the predicted distribution which is determined by inputs and model parameters. After training, the probability distribution could give insights into both point prediction and prediction uncertainty.

$$Loss = - \sum_{i=1}^N \sum_{t=0}^T \log p(z_{i,t} | \mu(\mathbf{h}_{i,t}), \sigma^2(\mathbf{h}_{i,t})) \quad (2.6)$$

In addition to the probabilistic layer, more approaches to uncertainty estimating: bayesian neural network (BNN), deep ensemble, and MC-Dropout are available in appendix A. Although BNN has advantages in the theoretical aspect, it requires a lot of effort in the training phase due to the large number of integrations. And in practice, MC-Dropout and deep ensemble cost high computation power in prediction.

## Quantifier

Above introduced uncertainty-aware deep models all produce prediction distributions. To extract uncertainty from predictive distribution or point predictions, a quantifier is necessary. For regression tasks, variance is the most common quantifier. In [27], the authors proposed to use distribution mean and distribution variance as prediction and uncertainty, respectively.

## Chapter 3

# Problem Formulation

Above introduced methods are implemented to the research problem, which revolves around error modeling for a tractor-semitrailer dynamics model, and the following detailed problem description gives a connection from the research problem to background knowledge.

First of all, a tractor-semitrailer model that provides simulation data is the research target. The vehicle model aims to quickly produce simulation signals for multiple research tasks in the realm of autonomous driving. However, error is also generated from the vehicle model, which makes simulation signals not reliable. In order to compensate for the errors generated by the vehicle model, the core problem of this thesis is to build an error model which could learn and predict errors for lateral dynamics. A valid error model that passes evaluation tests on test data is able to complement the vehicle model. The relation of vehicle, vehicle model, and error model is shown in figure 3.1.

A brief introduction to the vehicle model is presented in section 3.1. Different sources of gap between signals collected from vehicle and simulation data from the vehicle model are introduced in section 3.2. Tasks for building and validating error models are elaborated in section 3.3.

### 3.1 Volvo Dynamics Vehicle Model

This project is implemented on a dynamic vehicle model which is used to simulate a tractor semi-trailer. The nature of vehicle models is simulation of vehicle states composed of multiple signals given manoeuvres within a certain period under certain circumstances. The structure of a specific vehicle model, as well as the composition of the input and output, is determined by the purpose, fidelity, and application scenarios of the model. Figure 3.2 shows the abstracted structure of the truck model in this project.

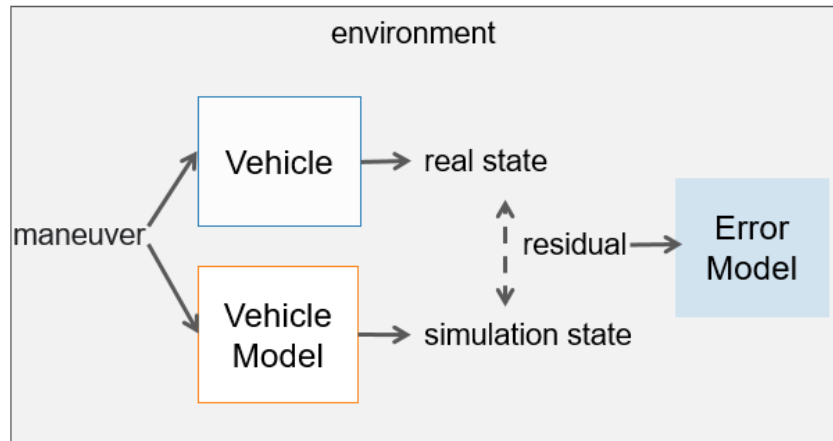


Figure 3.1: Vehicle model aims to producing dynamics signals that simulate real vehicle, however error is inevitable. Error model learns from vehicle model and real vehicle to make prediction for errors. Thus, error model could compliment vehicle model.

In this project, the internal principles of the model is not explored deeply, which can be considered as a reasonable black box model based on physical principles. It's certain that at each moment, the input of the model represents certain manoeuvre is denoted by a vector and is fed to the black box, and the corresponding output that represent simulation truck state is provided also as a state vector. Within the process of simulation, each component of the model is possible to introduce errors compared to the real system. More specifically, this project focus on model error for lateral dynamics of the truck. Thus, the goal of this project is to improve vehicle model simulation for yaw rotation rate, which is one of the representatives of lateral dynamics.

## 3.2 Vehicle Model Residual

To explore how accurately the vehicle model simulates the real system, a large amount of data from the model as well as the system is collected, covering a wide variety of operations.

Firstly, different manoeuvres are performed on the real truck, and state values of the truck are collected at discrete time steps using sensors. These data collected from sensors, so-called 'sensor data' later in this thesis, have noise introduced by sensors. After cleaning sensor data, dynamic inputs which are fully consistent with denoised sensor data are passed to vehicle model and the simulated data is

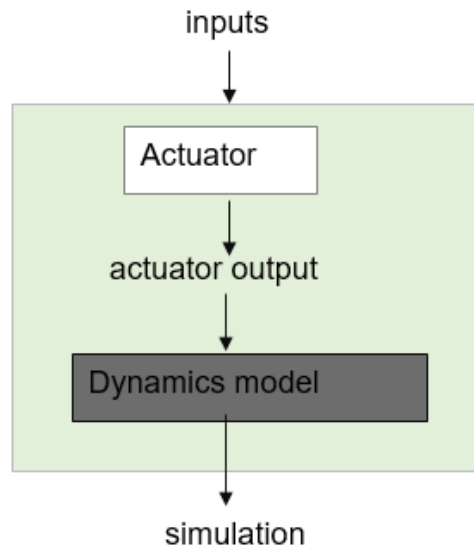


Figure 3.2: Vehicle model structure. A vehicle model is comprised of the controller, actuator, and dynamic model.

obtained.

The above two steps respectively result in sources of gaps between the simulation data and the real system, as shown in figure 3.3. First, there is gap between simulation data and real data, which is 'residual'. It's reasonable to consider two sources of residual, model error and disturbance. The former is caused by the imperfection of the vehicle model and is related to the inputs of the model, and its existence means that the model can be further optimized. The latter is produced by external situations that are not represented by model input. Second, the sensor generates noise when collecting data, resulting in inconsistency between the real vehicle dynamics and the sensor data, although sensor noise is relatively smaller than the residual. As introduced in section 2.2, only error is variant to vehicle model inputs. In other words, with data that is currently involved in a model, noise or disturbance cannot be avoided, but model error is possible to be reduced by improving vehicle models.

In summary, the difference between simulation data and sensor data comes from three aspects, sensor noise, model error, and disturbance. Since error is predictable, the project goal is to improve simulation results for yaw rotation rate by compensating model error.

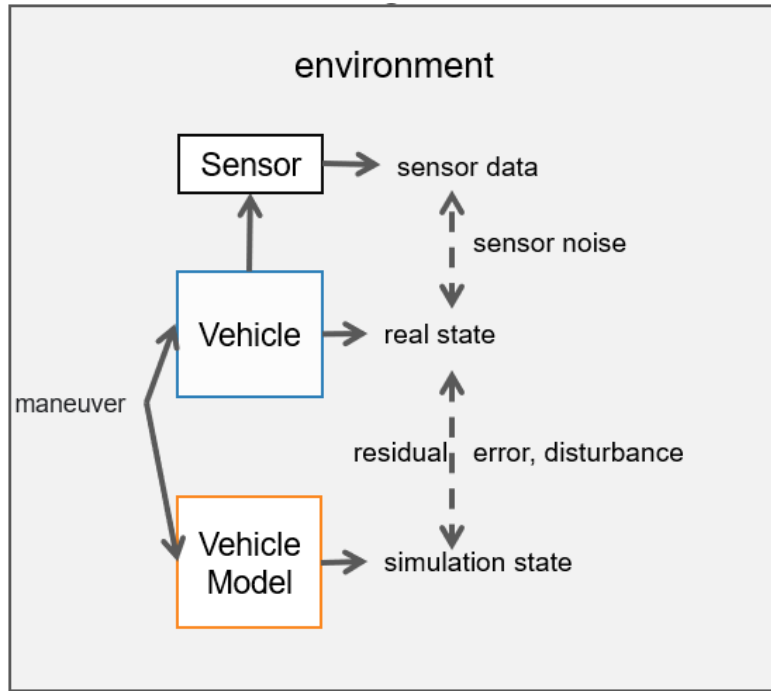


Figure 3.3: Source of residual and noise. Error and disturbance constitute residual, while noise is caused by sensor.

### 3.3 Error Model

Now that the vehicle model based on physics has simulation error, data-driven DL models are chosen as architecture of error models. DL cannot reduce error of vehicle model directly, but DL models can predict vehicle model error given sufficient data. Thus, to improve simulation results for yaw rotation rate, a possible method is to build error models based on DL and compensate model error with error model prediction.

Vehicle state is determined by the operation and past states, so the error models also predict the error from history states. Therefore, attempts are made to build error models based on RNN since RNN model architecture fully take use of history information. To make error model independent of real truck in prediction phase, vehicle model inputs and simulation results, which are available without real truck, are candidate information for error model construction, and vehicle simulation residual is learning target.

In addition to error prediction, uncertainty level about prediction results is also important knowledge since it indicates the reliability of applying error forecast to vehicle model. Thus, two categories of error models are established, the first

being point prediction models and the second being uncertainty-aware models, both expressed as a deep recurrent neural network. While point prediction models could only provide prediction for model error, uncertainty-aware models also tell the uncertainty about the prediction.

### 3.4 Error Model Evaluation and Validation

To prove that RNN-based error models have a good performance in predicting error, error model evaluation is necessary. Following the common idea for evaluating DL models, statistics metrics based on prediction residuals is selected, including Mean Square Error (MSE) and the correlation between the residuals and the past inputs. MSE visually describes how close the model is to the targets, while the latter metric can indicate whether the existing model residual contained error or disturbance. According to [13], only error is related to model inputs and disturbance cannot be learned from the data involved, thus, the value of the latter indicator reveals if the residual could be eliminated by further learning from model inputs. If the value is somewhat small, then it can be believed that the error model performs well at predicting error, and the remaining residuals are composed of disturbance and noise. Besides, log-probability of targets given model is applied to uncertainty-aware models.

To investigate if RNN-based error models are capable of complementing vehicle model, validation based on prediction results is designed for uncertainty-aware error model. Valid error models should meet the criteria set for the prediction results on the test data. Considering the purpose of validation is to guarantee the reliability of error models, a minimum bound for coverage rate of prediction interval to target is set.

In conclusion, the main purpose is to build error models that can predict simulation error for yaw rotation rate and compensate the simulation residual of truck model with error prediction results. With assumption that the truck model is able to simulate longitudinal signals with ignorable error, the research could be explained as modeling error for yaw rotation rate while longitudinal dynamics is fixed. To achieve this goal, the error models resorts to RNN models and are trained with vehicle simulation residual. Although simulation residual contain not only error but also noise and disturbance, noise will be reduced by frequency domain filtering in the data pre-processing stage, while the existence of disturbance is revealed by model evaluation metric based on the correlation between error prediction residual and inputs.

## Chapter 4

# Materials

Above mentioned research is applied to the vehicle model and data provided by Volvo Autonomous Solutions. Vehicle model is introduced in section 4.1. Data collecting and pre-process is in presented in section 4.2 and 4.3. An example of vehicle model simulation is displayed in section 4.4.

### 4.1 Tractor-semitrailer Dynamics Model

The physics-based tractor-semitrailer simulation model is abstracted as shown in figure 4.1. Neglecting its inner principles, this project pays attention to the data interaction. Firstly, the model inputs include 45 variables and some of them describe vehicle manoeuvres such as velocity, brake position, gear, payload fraction, and wheel angle, others represent the external environment including friction coefficient, height and slope of roads. Due to the lack of ability to monitor the external environment, related variables are replaced by constants. The friction coefficient is set as 0.7 while height and slope of roads are 0.

Given continuous input data, the vehicle model simulates thousands of output variables, and among them is the research target, yaw rotation rate, which describes lateral dynamics and means the rotation rate about the z-axis of the vehicle. In addition to yaw rotation rate, some other output variables including simulated acceleration and velocity, are also extracted. The simulation frequency is set as 100 Hz, which is not too low so that the simulation becomes coarse or too high so causing the vehicle model to slow down.

In summary, dynamic variables comprised of velocity, brake position, payload fraction, wheel angle, etc. are engaged in the simulation process with a simulation frequency at 100 Hz, and simulated variables including yaw rotation rate are extracted from simulation results.

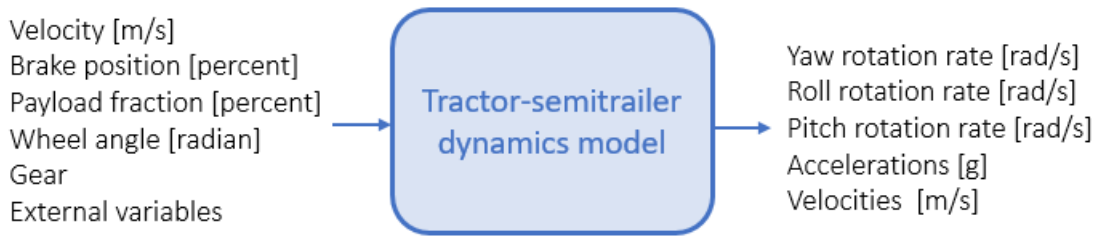
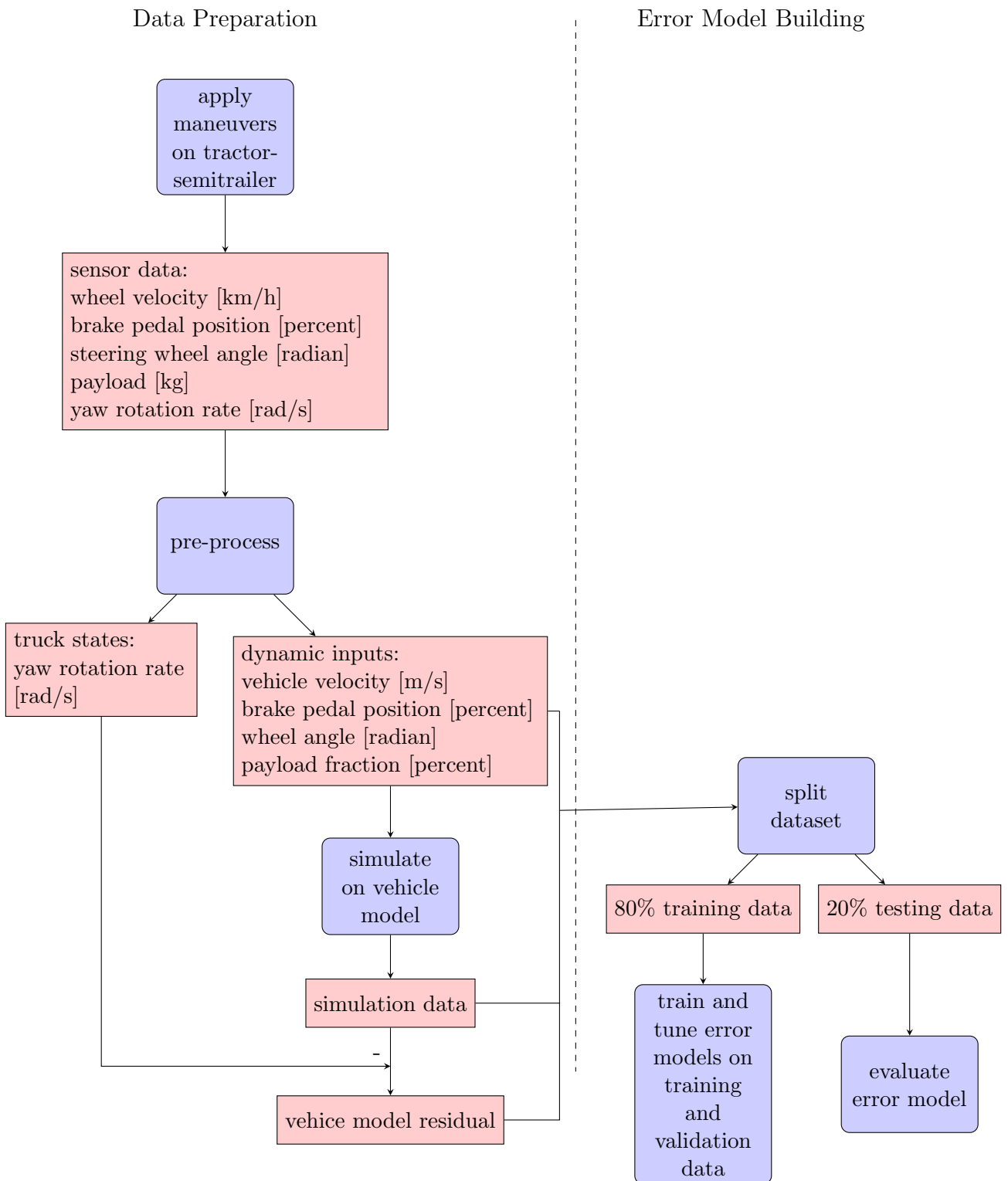


Figure 4.1: Abstract tractor-semitrailer dynamics model, a physics-based model that simulates tractor-semitrailer with tens of inputs variable and produces thousands of outputs.

## 4.2 Data Collection

The following flowchart shows the project flow from collecting data to training model, which follows below steps.

1. The first task is to collect and prepare data. Firstly, various manoeuvres are performed on the tractor-semitrailer, while sensors are collecting signals lasting for hours continuously. Depending on research tasks, at least wheel velocity, brake pedal position, steering wheel angle, payload, and yaw rotation rate are collected.
2. As discussed above, there is sensor noise between the sensor data and the real state of the vehicle, and this sensor noise is usually high-frequency noise and constant bias, so pre-process of the sensor data is implemented to eliminate sensor noise as presented in section 4.3.
3. After pre-processing, the input variables that denote manoeuvres are available to be fed into the vehicle model, and yaw rotation rate signals are also available to validate and evaluate models. The next step is to apply input variables to the vehicle model and collect simulated data as introduced in section 4.1.
4. Finally, pre-processed input variables, simulation data, and simulation residuals are normalized to  $[-1,1]$  interval and are involved in error model construction.



### 4.3 Sensor Data Pre-processing

The raw data collected from the sensors is generated by performing various maneuvers on vehicles for over 9,000 seconds continuously. Signals are collected at various sampling frequencies ranging from 0.5 Hz to 100 Hz. Table 4.1 lists some properties of signals involved in this project.

Table 4.1: Properties of collected signals .

Variable	Sampling rate	Mean	Variance
Steering wheel angle [rad]	100 Hz	0.12	1.48
Brake position [percent]	100 Hz	4.82	8.89
Gross weight [kg]	0.5 Hz	23488	624
Rear axle yaw rotation rate [d/s]	100 Hz	-0.80	4.62
Front left wheel velocity [km/h]	50 Hz	26.50	19.05
Front right wheel velocity [km/h]	50 Hz	26.51	19.02
Rear left wheel velocity [km/h]	50 Hz	26.43	19.02
Rear right wheel velocity [km/h]	50 Hz	26.44	19.00

First of all, the signals should be split into multiple series manually so that each series of signals starts from stationary status and is fed into the vehicle model individually, since simulation error is accumulated as time increases. Thus, the motionless period, i.e., when the vehicle is stationary lasting for at least 2 seconds, is marked. The front right wheel velocity is used to mark those motionless periods and a total of 46 motionless periods are detected as shown in figure 4.2, so all signals are separated into 47 series. In addition to splitting data series, motionless periods data is also auxiliary for subsequent frequency filtering.

Besides, gross weight is transformed to payload fraction by equation 4.1, where vehicle weight is 8932.3 kg and full load is 24000 kg. And steering wheel angle is transformed to road wheel angle by dividing transform ratio 18.6. After these basic process, the sensor data enters the frequency filtering stage to eliminate the noise brought by the sensor.

$$payload\_fraction = (gross\_weight - vehicle\_weight) / full\_load \quad (4.1)$$

#### Noise Reduction

Before implementing the filter, some supplementary information is collected to restrict filters and calibrate filtered signals to the correct range. This information, as

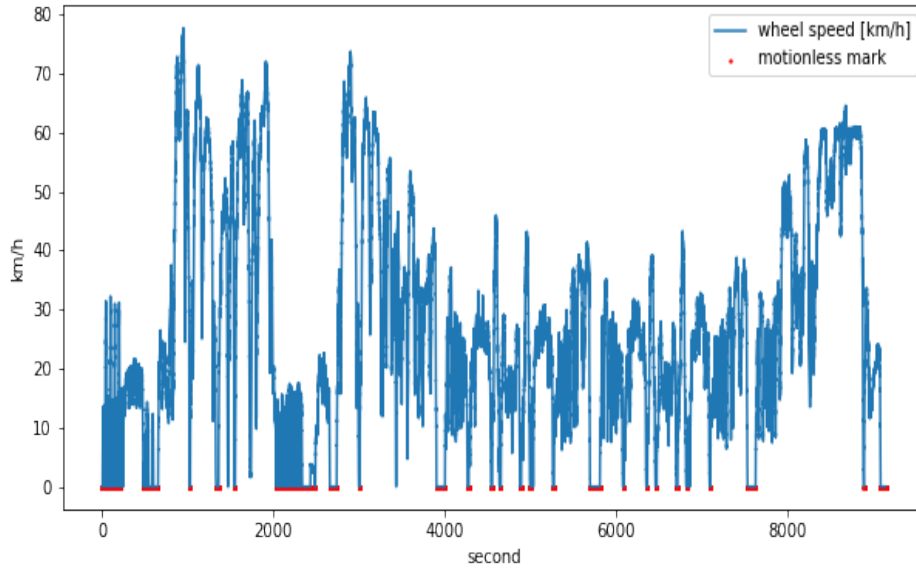


Figure 4.2: Wheel velocity and motionless periods marks.

shown in table 4.2, includes theoretical bound, and desired value during motionless periods.

Table 4.2: Supplementary information.

Signal	Minimum	Maximum	Stationary value
Wheel angle [rad]	-0.7	0.7	-
Brake position [percent]	0	100	-
Payload fraction [percent]	0	100	-
Yaw rotation rate [d/s]	-30	30	0
Wheel velocities [km/h]	0	100	0

Besides, as mentioned above, motionless period data visualization gives an insight into whether there is constant bias produced by sensors and whether the data is noised. For instance, figure 4.3 shows the signal of yaw rotation rate during motionless periods. Yaw rotation rate should be 0 within these periods, however, the collected data has an average bias at around -0.61. This figure also indicates that the signal is strongly fluctuating during motionless periods, which means the sensor brings high-frequency noise.

With the assistance of above information, filter type, filter cut-off value, and

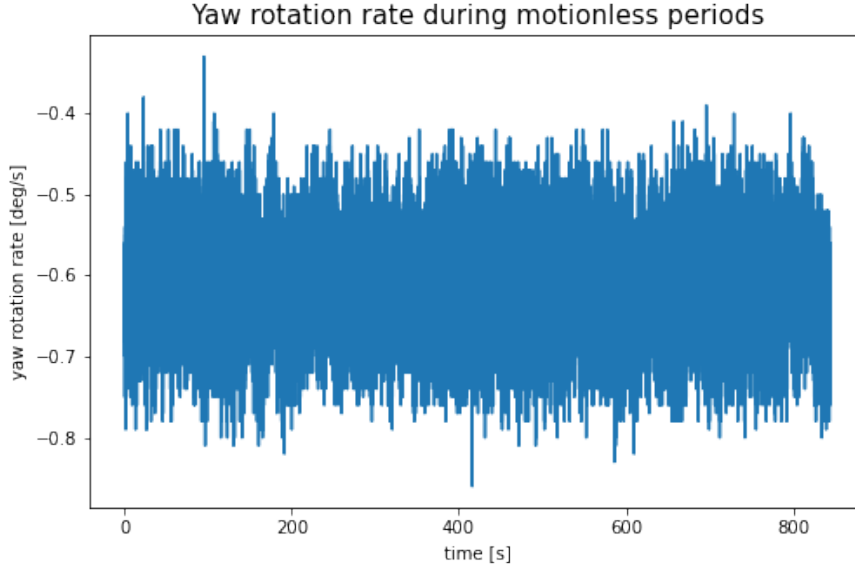


Figure 4.3: Yaw rotation rate during motionless periods. In theory, the mean value should be 0 but in fact is around -0.61. Meanwhile, the signal shows high frequency fluctuation.

bias value for each signal is decided as shown in table 4.3. Low-pass filtering is implemented to all variables excluding load weight, since the original signal of load weight has a low sampling frequency (0.5 Hz), and low-pass filters under 0.25 Hz could result in a huge offset. For variables including pedal position and steering wheel angle, in practice the frequency cannot exceed 5 Hz, thus, they are filtered with a cut-off at 5 Hz. The yaw rotation rate is affected by steering and road situation. From the steering aspect, the frequency of the yaw rotation rate change should not exceed 5 Hz, but considering the uncertainty caused by the road, the filtering is relaxed to 10 Hz. Wheel velocities are also filtered at 10 Hz due to the affect of road situation. Butter filters that have attenuation effects are used in this project, to reduce the effect of attenuation, the order of filters is set high in some sense, since the higher order of butter filters is, the higher the attenuation rate is[22].

Additionally, frequency analysis based on frequency spectrum is applied to assist in determining the cut-off value of filters. A frequency spectrum example is displayed in figure 4.4. As shown in the figure, 10 Hz is much high than the maximum frequency of yaw rotation rate signal. Thus, low-pass filter at 10 Hz would lose slight information.

A clipped segment from denoised results of yaw rotation rate is presented in

Table 4.3: Filter parameters.

Signal	Filter type	Cut-off	Filter order	Bias
Wheel angle [rad]	low-pass filtering	5 Hz	9	0
Brake position [percent]	low-pass filtering	5 Hz	9	0
Wheel velocities [km/h]	low-pass filtering	10 Hz	9	0
Yaw rotation rate [d/s]	low-pass filtering	10 Hz	9	+0.61

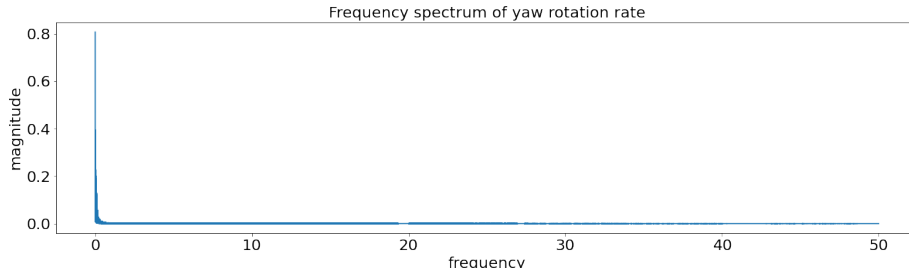


Figure 4.4: Frequency spectrum of yaw rotation rate, and the frequency hardly exceeds 10 Hz.

figure 4.5. As shown in the figure, yaw rotation rate was filtered to be smoother and was also corrected from constant bias.

After noise reduction, the obtained signal is split into shorter series at motionless time steps. Linear interpolation is applied to signals that have under 100 Hz sampling rate, to make all signals have the same sampling rate. Table 4.4 presents properties of signals after above pro-process.

Table 4.4: Properties of denoised signals.

Signal	Mean	Variance
Wheel angle [rad]	0.01	0.08
Brake position [percent]	4.86	8.90
Payload fraction [percent]	0.61	0.03
Rear axle yaw rotation rate [d/s]	-0.20	4.62
Front right wheel velocity [km/h]	26.51	19.02
Front left wheel velocity [km/h]	26.50	19.05
Rear left wheel velocity [km/h]	26.43	19.02
Rear right wheel velocity [km/h]	26.44	19.00

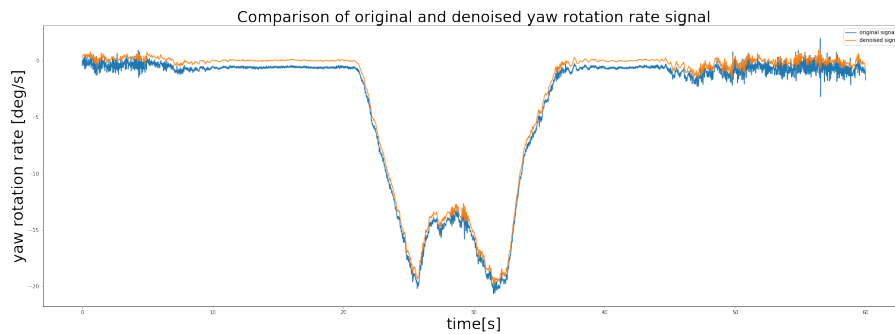


Figure 4.5: Comparison of filtered and original yaw rotation rate signal. Filtering makes signal smoother, and the constant bias is removed.

## 4.4 Vehicle Model Simulation

As presented in the above flowchart, pre-processed signals are fed into the vehicle model to simulate yaw rotation rate. To avoid accumulating error, each series is simulated individually and does not affect each other. Figure 4.6 shows the simulation result as well as the residual for yaw rotation rate for a randomly chosen series.

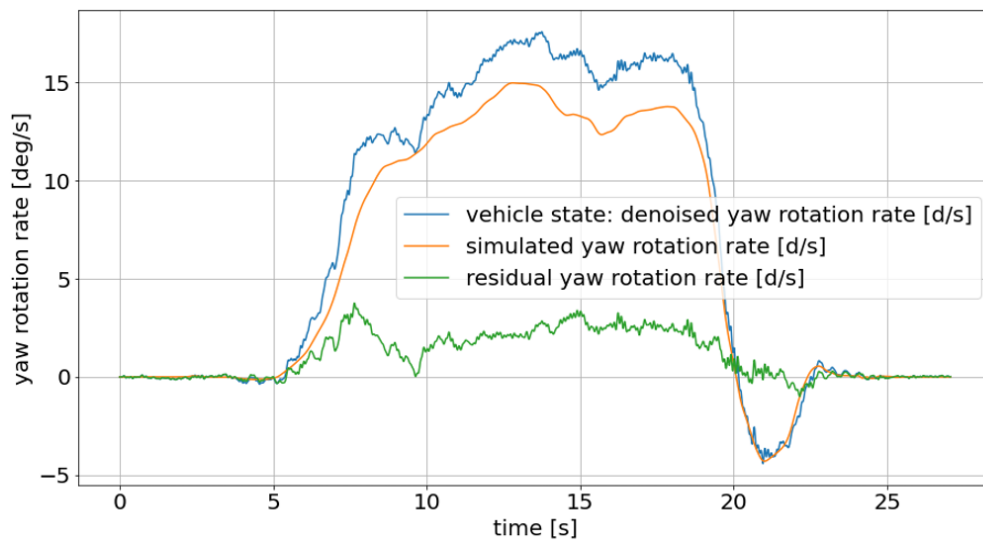


Figure 4.6: Simulation result and residual for yaw rotation rate. Apparently simulation result is smoother, since sensor data contains not only error, but also disturbance and noise.

# Chapter 5

## Methods

To fully utilize the data, following experiments for building a well-performing error model are designed. Model architecture, model training, evaluation, and validation methods will be introduced in this chapter.

### 5.1 Error Model Architecture

As elaborated in chapter 2, point prediction models only provide prediction to target without other information, while uncertainty-aware models could not only give predictions, but also offer insights into the uncertainty about the predictions. Two types of RNN error models are designed, and they have similar basic architecture but different last layer. The difference in last layer makes one point prediction type and another to be uncertainty-aware.

#### Point Prediction Model

The point prediction model architecture is shown in figure 5.1. The hidden layer is a stack of 2 LSTM layers. A linear layer follows the hidden layers and produces error prediction.

In the architecture,  $x_{i,t}$  denotes the  $t^{th}$  inputs vector of the  $i^{th}$  series. During the training process, at time  $t$ , inputs vector  $x_{i,t}$  in conjunction with previous hidden state  $h_{i,t-1}$  decides  $h_{i,t}$  at hidden layers, and then  $h_{i,t}$  is used to calculate final output at the linear layer. Target  $y$  is not engaged in feed-forward calculation, but contributes in BP process.

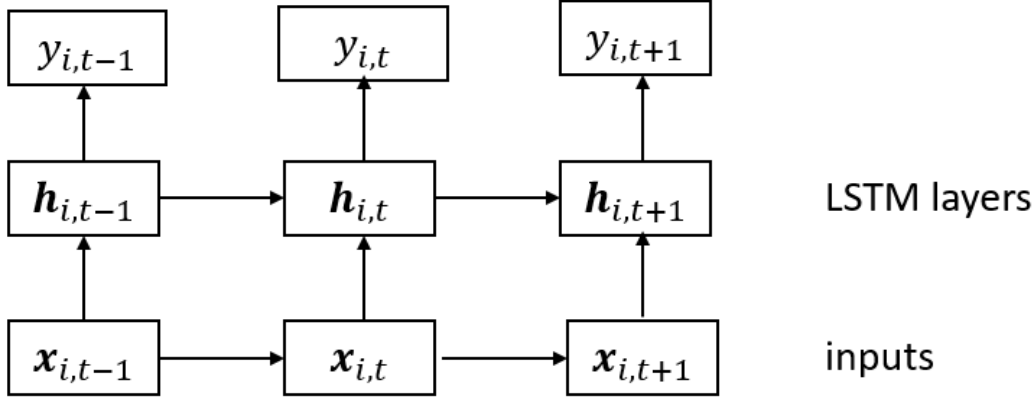


Figure 5.1: Point prediction model architecture, comprised by LSTM layers and a linear layer.

## Uncertainty-aware Model

Inspired by DeepAR [19], a probabilistic layer, which is capable of providing probability distribution for target over possible value, is added to LSTM cell stack to form an uncertainty-aware model. The architecture shown in figure 5.2 is comprised of a stack of 2 LSTM layers, followed by a linear layer which produces distribution parameters.

Calculation logistics is the same with above point prediction model. Differently, distribution parameters are predicted by the uncertainty-aware model and during the prediction process, prediction  $\hat{y}$  could be extracted from the predicted distribution. Thanks to nature of the data, it is believed that disturbance and noise, which is source of prediction uncertainty, follows Normal distribution. Thus, normal distribution is selected as the predictive distribution type to match the statistical properties of the data. Following deepAR, the parameter  $\mu$  and  $\sigma$  is formed as equation 2.4 and 2.5. As introduced in section 2.4, variance of predictive distribution is a common quantifier for uncertainty, and prediction interval  $[\mu_{i,t} - z\sigma_{i,t}, \mu_{i,t} + z\sigma_{i,t}]$  with certain confidence is available from predictive distribution.

## 5.2 Training Inputs and Targets

As discussed in section 2.2, vehicle model input variables and vehicle model simulation results are involved in error model training. So totally 18 variables are inputs for error model, as listed in table 5.1. Regarding error model targets, as

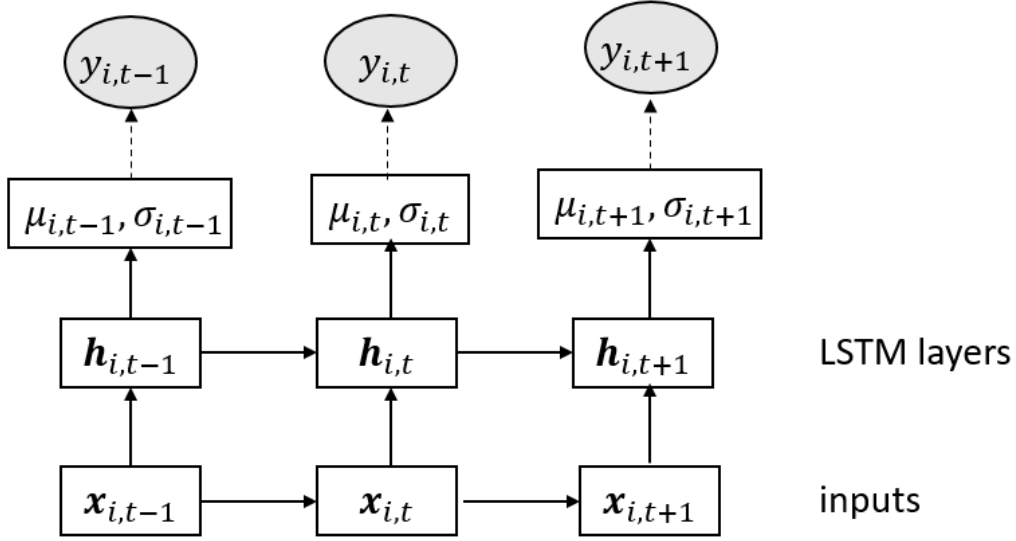


Figure 5.2: Uncertainty-aware model architecture, comprised by LSTM layers and a linear probabilistic layer.

introduced in chapter 4, residual between pre-processed sensor data and vehicle model simulation is available. Considering that three sources of gap is inseparable, the residual is used as learning target for error models, although it contains not only model error, but also disturbance and some sensor noise that is not completely eliminated. According to [13], noise and disturbance is unrelated to model input, thus, an ideal error model can learn model error but not disturbance or noise from the learning target.

### 5.3 Loss Functions and Evaluation Metrics

Following deepAR, the loss function for uncertainty-aware model is negative log-likelihood. As shown in equation 5.1, minimizing the loss function equals to maximizing the sum of probability of target  $y_{i,t}$  given normal distribution  $\mathcal{N}(\mu_{i,t}, \sigma_{i,t}^2)$ , over series and time steps. This loss function takes accuracy of the target prediction into consideration and also reduce the associated uncertainty as much as possible. For point prediction model, MSE as equation 2.1 is chosen as the loss function.

$$Loss = - \sum_{i=1}^N \sum_{t=0}^T \log p(y_{i,t} | \mu(\mathbf{h}_{i,t}), \sigma^2(\mathbf{h}_{i,t})) \quad (5.1)$$

Table 5.1: Error model input variables.

Source	Variable	Unit
Sensor signal	Vehicle velocity	m/s
	Brake pedal position	percent
	Wheel angle	radian
	Payload fraction	percent
Vehicle model simulation	Front axle yaw rotation rate	d/s
	Rear axle yaw rotation rate	d/s
	Front axle roll rotation rate	d/s
	Rear axle roll rotation rate	d/s
	Front axle pitch rotation rate	d/s
	Rear axle pitch rotation rate	d/s
	Front axle longitudinal acceleration	g
	Rear axle longitudinal acceleration	g
	Front axle lateral acceleration	g
	Rear axle lateral acceleration	g
	Front axle longitudinal velocity	Km/h
	Rear axle longitudinal velocity	Km/h
	Front axle lateral velocity	Km/h
	Rear axle lateral velocity	Km/h

To measure the performance of regression results, MSE is applied to both uncertainty-aware models and point prediction models. For uncertainty-aware model, MSE is implemented on targets and predicted  $\mu$ . Additionally, negative log-likelihood is applied to uncertainty-aware models. Besides, Pearson correlation between error model prediction residual and past inputs as shown in equation 5.2 is another metric, which reveals the model ability to learn errors from vehicle model residuals. In equation 5.2,  $\phi_\tau$  and  $\epsilon_\tau$  is vector comprised of specific input variable and error model residual respectively, and there is  $\tau$  seconds shift between them. Thus, the correlation between the past input variable and error model residual is expressed as correlation between  $\phi_\tau$  and  $\epsilon_\tau$ .

$$\begin{aligned}
\rho_\tau(\mathbf{x}, \mathbf{y} - \hat{\mathbf{y}}) &= \rho(\phi_\tau, \epsilon_\tau) \\
\phi_\tau &= [x(1), x(2), \dots, x(T - \tau)] \\
\epsilon_\tau &= [\hat{y}(\tau + 1) - y(\tau + 1), \hat{y}(\tau + 2) - y(\tau + 2), \dots, \hat{y}(T) - y(T)]
\end{aligned} \tag{5.2}$$

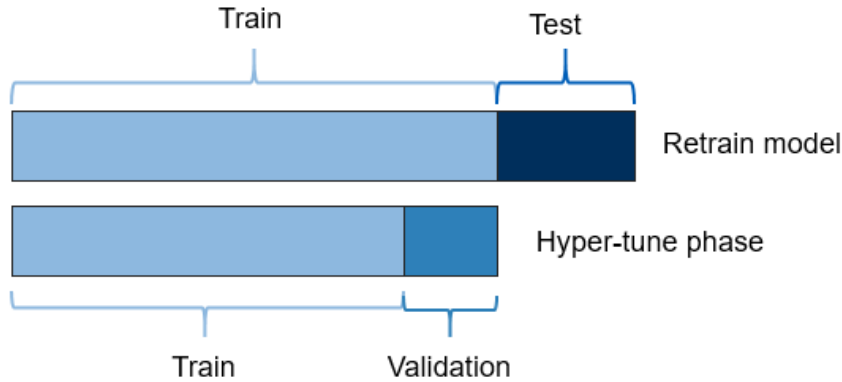


Figure 5.3: Data separation.

## 5.4 Hyper-parameter Configuration

To select best hyper-parameter configuration, the series is split to training and testing data at a ratio of 8:2, and during the hyper-tune phase, model is trained with 80 percent training data while validation is applied to the last 20 percent of training data to search over optional hyper-parameter values. The best configuration, with respect to above evaluation metrics, is then used to retrain a model with the whole training data and the model will be evaluated and validated on test data. The training, validating, and testing data separation is shown as figure 5.3. With assumption that test data follows distribution of unseen data, the final evaluation results could be seen as the estimated performance on future data. Due to computational power limitation, only following hyper-parameters are tuned:

Table 5.2: Hyper-parameters for RNN based error model.

Hyper-parameter	Search space	Description
Batch size	[8,16,32,64]	# of samples propagated to the network at once.
Epochs	1-50	# of passes of the entire training data.
TBPTT *	[128,256]	Truncated back-propagation through time.
Learning rate (LR)	[0.01,0.05,0.1,0.2]	Step size at each iteration while moving towards a minimum of the loss function.

\* TBPTT, is a training algorithm for recurrent neural networks where "the sequence is processed one time step and every  $k_1$  time steps, it runs BPTT for  $k_2$  time steps" [24], in this thesis same TBPTT value is chosen for  $k_1$  and  $k_2$ .

## 5.5 Validation Criteria

Evaluation results tell if the error model has good performance to predict error, while validation results represent if a error model is accepted and will be used to compliment the vehicle model. A good error model is expected to provide probability distribution of which real target is included in 99% prediction interval, as shown in formula 5.3. In this project, error models that provide over 80% samples with useful prediction interval are considered as valid models. Thus the validation criteria requires 99% prediction interval coverage rate (CR) on test data is not less than 0.8, as shown in formula 5.4. Meeting validation requirements means reliability of error models.

$$\mu_t - 2.58 \sigma_t \leq y_t \leq \mu_t + 2.58 \sigma_t \quad (5.3)$$

$$|\{t | \mu_t - 2.58 \sigma_t \leq y_t \leq \mu_t + 2.58 \sigma_t\}| \geq 0.8 T \quad (5.4)$$

## Chapter 6

# Results

The best hyper-parameter configuration for the point prediction model and the uncertainty-aware model, along with corresponding performance on the validation set, is presented in table 6.1.

Table 6.1: The best hyper-parameter configuration.

	Batch size	TBPTT	LR	Epochs	Performance
Point prediction model	8	128	0.01	41	0.11 (MSE)
Uncertainty-aware model	8	128	0.1	48	-0.70 (NLL)

Retraining two types of error models with above hyper-parameters on all training data and applying the models to testing data, prediction results of error models are available. Results for a random chosen data series as shown in 6.1, and prediction interval in figure 6.1(b) is 99% prediction interval. Then, error prediction from two error models is combined with vehicle model simulation, and results for the series are shown in figure 6.2.

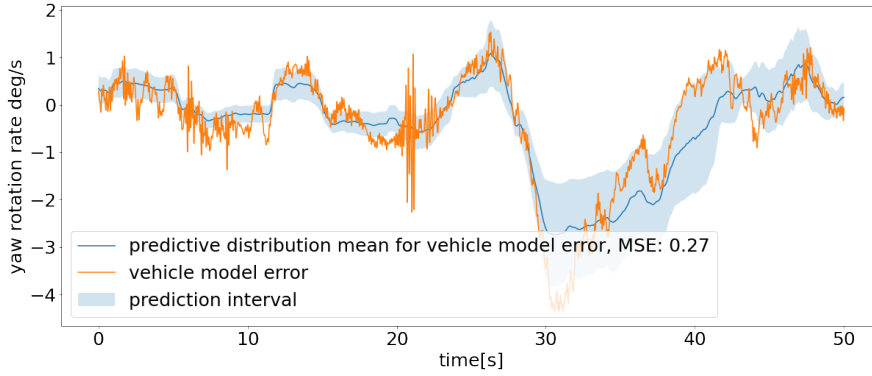
Besides, evaluation results on test data for the original vehicle model and error models are presented in table 6.2.

Table 6.2: Error model evaluation results.

	Vehicle model	Point prediction model	Uncertainty-aware model
MSE	0.5	0.13	0.13
Average NLL	-	-	-0.62
99% prediction interval CR	-	-	0.5
$\sum_{\tau \in D}  \rho_{\tau}(\textit{payload fraction}) $	0.075	0.002	0.010
$\sum_{\tau \in D}  \rho_{\tau}(\textit{brake pedal position}) $	0.015	0.050	0.014
$\sum_{\tau \in D}  \rho_{\tau}(\textit{wheel angle}) $	0.086	0.111	0.098
$\sum_{\tau \in D}  \rho_{\tau}(\textit{wheel speed}) $	0.014	0.014	0.070



((a)) Prediction of point prediction model.



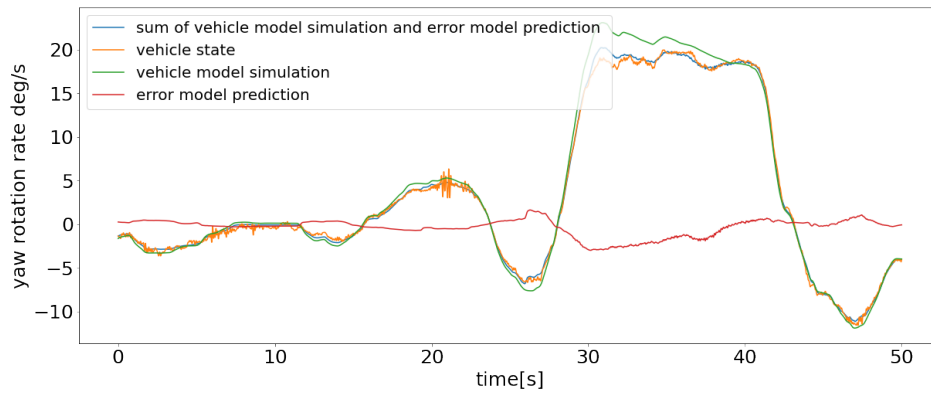
((b)) Prediction of uncertainty-aware model.

Figure 6.1: Error models prediction results.

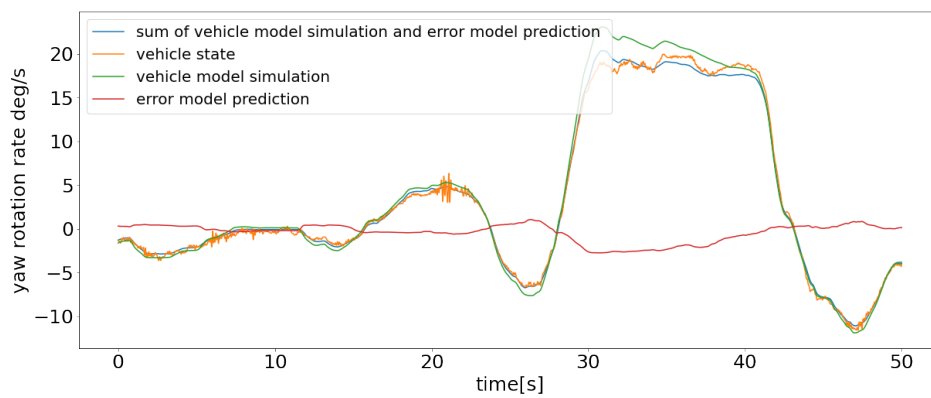
Additionally to MSE and average of NLL, the average of  $|\rho_\tau(\mathbf{x})|$  for error models while  $\tau \in D = \{0, 0.01, 0.02, \dots, 9.99, 10.00\}$  is calculated and compared with the results of vehicle model as shown in figure 6.3.

Regarding the prediction interval coverage rate for the uncertainty-aware error model, figure 6.4 shows the absolute value of prediction residual against the predictive standard deviation of the uncertainty-aware error model on testing data. Validation requirement in formula 5.4 equals to formula 6.1, which corresponds to samples that are under the red line in figure 6.4. The proportion of samples under three lines are respectively, 0.49, 0.39, and 0.24, which represent the coverage rate of predictive interval  $[\mu_t - z\sigma_t, \mu_t + z\sigma_t]$  to  $y_t$  while  $z$  is 2.58, 1.96, and 1.15 respectively. In other words, the coverage rate of 99% prediction interval is 0.49.

$$|y_t - \hat{y}_t| \leq 2.58 \sigma_t \quad (6.1)$$



((a)) Complementing vehicle model simulation with prediction of point prediction error model.



((b)) Complementing vehicle model simulation with prediction of uncertainty-aware error model.

Figure 6.2: Combination of error models prediction and vehicle model simulation.

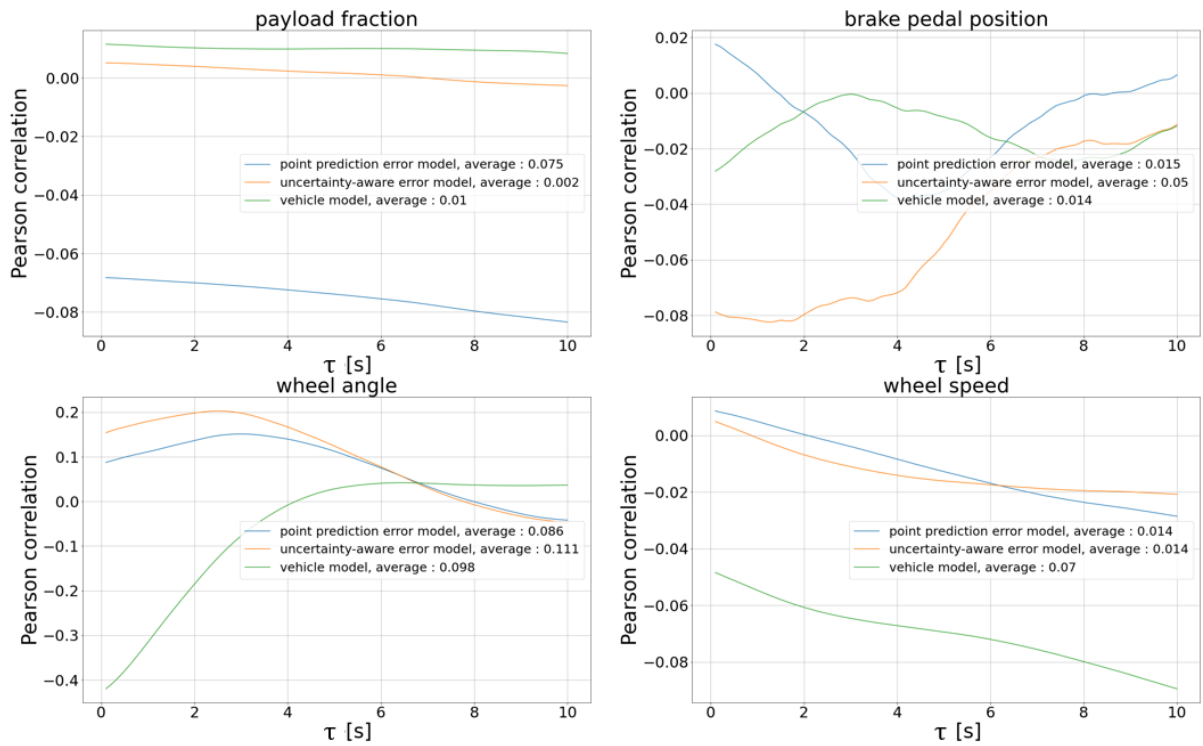


Figure 6.3: Pearson correlation between model residual and input variables over past 10 seconds.

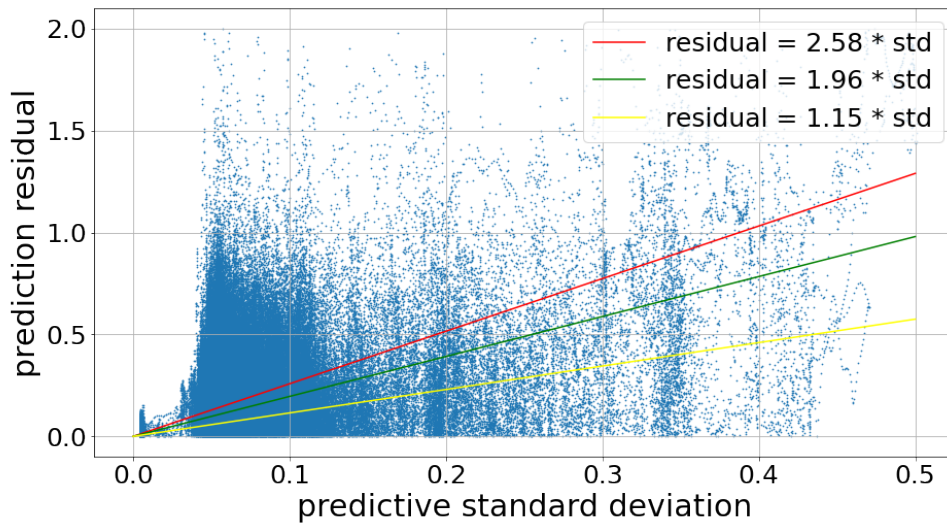


Figure 6.4: Prediction residual to predictive standard deviation.

# Chapter 7

## Discussion

In this chapter, first, an explanation for above experiment results is given in section 7.1. Then, some drawbacks and possible improvements for error models are specified in section 7.2. Finally, potential applications of vehicle model error models are put forward in section 7.3.

### 7.1 Explanation of Experiment Results

Firstly, the evaluation results indicate the power of deep learning-based error models from the following aspects.

1. Compared to the mean square of vehicle model simulation residual on testing data, 0.5, error models produce lower MSE at 0.13.
2. As shown in figure 6.3, two error models both produce a significantly lower correlation between residual and wheel speed than the vehicle model. Besides, as vehicle model residual has high correlations with wheel angle from 2 seconds ago to the present, error models are more satisfactory. Although for other input variables, error models do not perform better than the vehicle model concerning correlation, the correlation values remain at an acceptable level.

According to the performance of two error models with respect to MSE and residual-input correlation, it is believed that deep learning-based models are able to predict vehicle model errors for the truck model.

Although the complementing of error model prediction offset residual between vehicle model and real state to some extent as presented in figure 6.2, the uncertainty-aware error model does not meet validation criteria. As broaden predictive interval, the coverage rate increases but interval information becomes pointless, and

the opposite way produces non-accurate intervals. Thus, the uncertainty-aware model fails to provide accurate yet useful predictive intervals. Unfortunately, the uncertainty-aware error model is not accepted.

In summary, deep learning is applicable for predicting vehicle model errors. But the uncertainty-aware model cannot be put into use as it fails in validation tests.

Additionally, some hyper-parameters show preferences to specific values in the experiments, and the following is some guesstimate of the reasons:

1. As the learning rate increases from 0.001 to 0.1, the converge becomes faster. But the performance on the validation set shows fluctuation when the learning rate is larger than 0.1.
2. A large batch size saves training time while a small one is more likely to reach the global optimum. For given data, batch size prefers the lowest choice, 8. The reason might be that, when splitting data series into batches, the connection between batches is not learned. Thus, the lower the batch size, the less information is lost.

## 7.2 Shortcomings and Future Plans

First of all, the validation test is designed for error models that could produce predictive distribution. Thus, a problem is how to define the reliability level for point-prediction error models. A possible method is to apply MC-Dropout as introduced in A. MC-Dropout makes it possible for point-prediction models in any architecture to produce multiple predictions for one target, and predictive intervals is then available based on predictions. Thus, the predictive interval-based validation method will be applicable.

Secondly, the error models are not autoregressive. So the prediction must start from the beginning of a series. To save prediction time and broaden application scenarios, autoregression such as DeepAR is an optional choice. Simply put, during training the target value at the previous time step is used as input, and for prediction, the present prediction value is drawn and used to predict the next target. In this way, error models can become autoregressive and prediction can be started at any time step.

Finally, due to technical limitation, the number of LSTM layers, number of neurons, and the dropout rate is not tuned. It could be believed the error model can get better if more efforts are made to hyper-tune.

### 7.3 Potential Application of Error Models

In addition to complementing vehicle models, error models are also potential in the following scenarios.

1. Vehicle data collection. Physics-based vehicle models perform badly under certain circumstances, and collecting cases for those situations is useful to analyze and improve the vehicle model. Thus, applying an error model while driving the vehicle could give information about the real-time predictive error, and this is an indicator for collecting desired data. Besides, error prediction uncertainty is also helpful in special case collecting. Assuming that environment is modeled in error models, which means the uncertainty brought by the unknown environment is eliminated and the remaining prediction uncertainty comes from the data itself, then those samples that produce high uncertainty are worthy of studies.
2. Vehicle model evaluation. For specific vehicle models, the corresponding error model is an intuitive way of evaluation. More specifically, the high the error prediction, the worse the vehicle model on given data.

## Chapter 8

# Conclusions

In summary, the error model is put forward in this thesis as a solution to the problem of vehicle model simulation error, which learns from the residual of vehicle models and predicts the error given necessary information. Based on LSTM and probabilistic layer, point-prediction error model and uncertainty-aware error model architecture is designed. To investigate the possibility of deep learning in error modeling, evaluation methods based on residual analysis are designed. To check the reliability of error models, validation criteria for uncertainty-aware models is set.

The above error modeling methods are applied to a tractor-semitrailer dynamics model and training data is collected from corresponding trucks. The results show that deep learning is potential in error modeling. However, the uncertainty-aware model fails in the validation test. There is still a lot of space to improve error models, but this work is believed to be a meaningful attempt.

# Bibliography

- [1] BUNTINE, W. L. Bayesian backpropagation. *Complex Systems* 5 (1991), 603–643.
- [2] DER KIUREGHIAN, A., AND DITLEVSEN, O. Aleatory or epistemic? does it matter? *Structural Safety* 31, 2 (2009), 105–112.
- [3] DINEFF, A. M. Validation of tractor-semitrailer vehicle model based on bayesian hypothesis testing. Master’s thesis, Chalmers University of Technology, 2021.
- [4] DUPOND, S. A thorough review on the current advance of neural network structures. *Annual Reviews in Control* 14 (2019), 200–230.
- [5] FDELOCHE. A schema for lstm neural network architecture, June 2017. [Online. Available: [https://commons.wikimedia.org/wiki/File:Long\\_Short-Term\\_Memory.svg](https://commons.wikimedia.org/wiki/File:Long_Short-Term_Memory.svg)].
- [6] FDELOCHE. Structure of rnn, June 2017. [Online. Available: [https://commons.wikimedia.org/wiki/File:Recurrent\\_neural\\_network\\_unfold.svg](https://commons.wikimedia.org/wiki/File:Recurrent_neural_network_unfold.svg)].
- [7] GAL, Y., AND GHAHRAMANI, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning* (2016), PMLR, pp. 1050–1059.
- [8] GERS, F. A., SCHMIDHUBER, J., AND CUMMINS, F. Learning to forget: Continual prediction with lstm. *Neural Computation* 12, 10 (2000), 2451–2471.
- [9] HOCHREITER, S., AND SCHMIDHUBER, J. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.
- [10] LAKSHMINARAYANAN, B., PRITZEL, A., AND BLUNDELL, C. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in Neural Information Processing Systems* 30 (2017).

- [11] LANDRY, M., MALOUIN, J.-L., AND ORAL, M. Model validation in operations research. *European Journal of Operational Research* 14, 3 (1983), 207–220.
- [12] LJUNG, L. *Model validation and model error modeling*. Linköping University Electronic Press, 1999.
- [13] LJUNG, L., AND HJALMARSSON, H. *System identification through the eyes of model validation*. Linköping University, 1995.
- [14] MACKAY, D. J. A practical bayesian framework for backpropagation networks. *Neural Computation* 4, 3 (1992), 448–472.
- [15] NEAL, R. Bayesian learning via stochastic dynamics. *Advances in Neural Information Processing Systems* 5 (1992).
- [16] NIELSEN, M. A. *Neural networks and deep learning*, vol. 25. Determination press San Francisco, CA, USA, 2015.
- [17] PARRA, A., CAGIGAS, D., ZUBIZARRETA, A., RODRÍGUEZ, A. J., AND PRIETO, P. Modelling and validation of full vehicle model based on a novel multibody formulation. In *IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society* (2019), vol. 1, IEEE, pp. 675–680.
- [18] RUMELHART, D. E., HINTON, G. E., AND WILLIAMS, R. J. Learning internal representations by error propagation. Tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.
- [19] SALINAS, D., FLUNKERT, V., GASTHAUS, J., AND JANUSCHOWSKI, T. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting* 36, 3 (2020), 1181–1191.
- [20] SARGENT, R. G., AND BALCI, O. History of verification and validation of simulation models. In *2017 Winter Simulation Conference (WSC)* (2017), IEEE, pp. 292–307.
- [21] SCHLESINGER, S. Terminology for model credibility. *Simulation* 32, 3 (1979), 103–104.
- [22] SINGH, P. Cutoff frequency calculator, June 2022. [Online. Available: <https://www.omnicalculator.com/physics/cutoff-frequencywhat-is-cutoff-frequency-cutoff-frequency-definition>].

- [23] SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., SUTSKEVER, I., AND SALAKHUTDINOV, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.
- [24] SUTSKEVER, I. *Training recurrent neural networks*. University of Toronto Toronto, ON, Canada, 2013.
- [25] TAKKOUSH, M. Formal model validation by reachset conformance between low and high order tractor semitrailer vehicle models. Master’s thesis, Chalmers University of Technology, 2020.
- [26] TISHBY, N., LEVIN, E., AND SOLLA, S. A. Consistent inference of probabilities in layered networks: Predictions and generalization. In *International Joint Conference on Neural Networks* (1989), vol. 2, IEEE New York, pp. 403–409.
- [27] WEISS, M., AND TONELLA, P. Fail-safe execution of deep learning based systems through uncertainty monitoring. In *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)* (2021), IEEE, pp. 24–35.

# Appendix A

## First appendix

### A.1 Uncertainty-aware Algorithms

#### Bayesian Neural Networks

History of Bayesian Neural Networks (BNN) dates back to 1980s with paper [26] [1]. Instead of scalar, BNN has a probabilistic weight vector. A comparison between conventional DNN and BNN is presented in figure A.1. While conventional DNN attempts to find 'optimal' scalar weights, BNN finds posterior probability distribution over weight vector in the form of a probability density function. To get the posterior distribution over weights, firstly a prior distribution of weights  $P(\mathbf{w})$  should be defined. Then, given training case  $\mathbf{D} = \{(x_1, y_1), \dots, (x_n, y_n)\}$ , the posterior distribution over weight could be calculated using Bayesian Theorem as shown in equation A.1.

$$P(\mathbf{w}|\mathbf{D}) = \frac{P(\mathbf{D}|\mathbf{w}) \times P(\mathbf{w})}{P(\mathbf{D})} = \frac{P(\mathbf{D}|\mathbf{w}) \times P(\mathbf{w})}{\int P(\mathbf{D}|\mathbf{w}) \times P(\mathbf{w})d\mathbf{w}} \quad (\text{A.1})$$
$$P(\mathbf{w}|\mathbf{D}) \propto P(\mathbf{D}|\mathbf{w}) \times P(\mathbf{w})$$

Correspondingly, point predictions of BNN for unseen sample  $(x, y)$  are made by "averaging the outputs obtained with all possible weight vectors, with each contributing in proportion to its posterior probability" [15] as shown in equation A.2, where  $f(x, \mathbf{w})$  represents the calculation of the network.

$$\hat{y} = \int f(x, \mathbf{w}) \times p(\mathbf{w}|\mathbf{D})d\mathbf{w} \quad (\text{A.2})$$

The predictive posterior distribution of output  $y$  could be obtained through equation A.3, where  $P(y|x, \mathbf{w})$  is the probability distribution which is in a predefined distribution type and parameterized by network's result given input  $x$  and

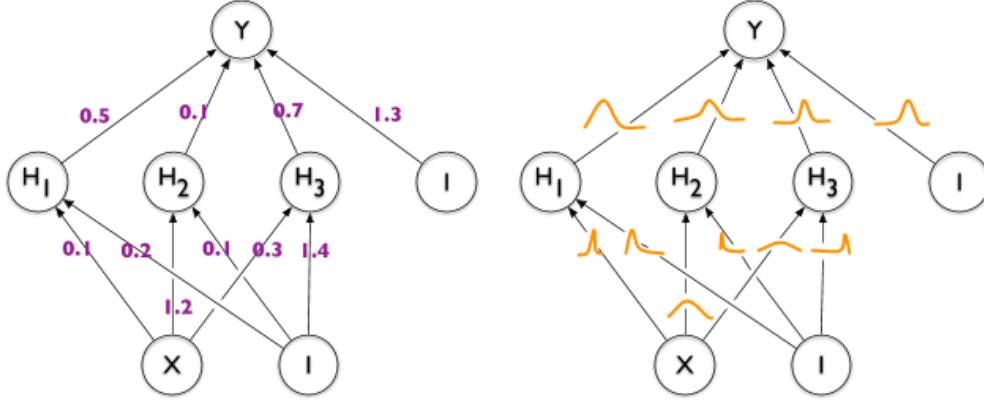


Figure A.1: Conventional DNN and BNN. The scalar edge weights of conventional DNN models is replaced by posterior distributions. Bayesian inference is the basis for calculating these posteriors.

weight  $\mathbf{w}$ . For instance, in [15] the target output  $y$  is assumed to be equal to  $f(x, \mathbf{w})$  plus Gaussian noise of standard deviation  $\sigma$ , thus, the network defines the conditional probability for output as equation A.4:

$$P(y|x, \mathbf{D}) = \int P(y|x, \mathbf{w}) \times p(\mathbf{w}|\mathbf{D}) d\mathbf{w} \quad (\text{A.3})$$

$$P(y|x, \sigma, \mathbf{w}) \propto \exp\left(\frac{-(y - f(x, \mathbf{w}))^2}{2\sigma^2}\right) \quad (\text{A.4})$$

One of the advantages of probabilistic weights is avoiding overfitting which is a common problem for conventional DNN [15]. Another benefit is that BNN provides probabilistic outputs which allow a "statistically well-founded uncertainty quantification" [27].

However, despite these advantages, BNN is not widely used due to computation difficulties brought by the large number of integrations, especially for predictive output distribution. Efforts have been made to solve this problem. In [15], integration is approximated by randomly sampling weights and averaging the prediction, instead of integrating over all possible weights. The sampling could be obtained using Markov Chain. Besides, Hamiltonian Monte Carlo (HMC), a sampling algorithm that makes use of gradients, is used to improve sampling efficiency. In [14], the posterior distribution of the weight vector is assumed to be a Gaussian, in order to simplify the integration.

In summary, BNN is theoretically well-founded regarding providing predictive probability distribution and measuring uncertainty. However, it is not technically feasible under some circumstances.

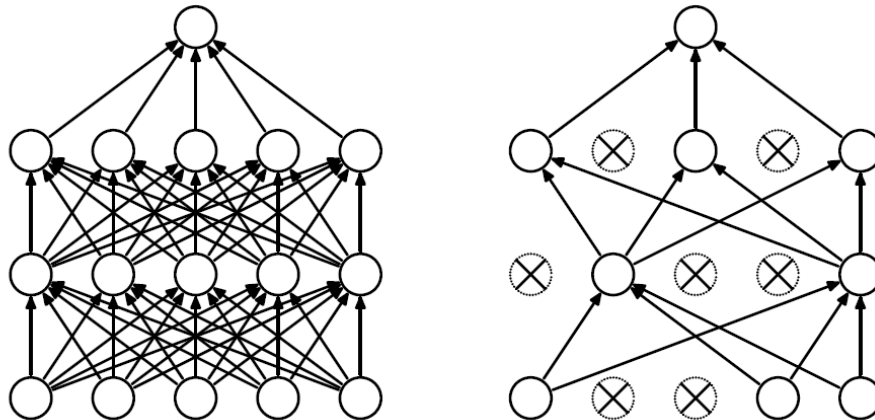


Figure A.2: A dropout neural network. Left: A standard neural network. Right: A neural network applying dropout.

## Monte-Carlo Dropout

Srivastava et al. proposed dropout in 2014 as a regularization approach applied on neural networks inspired by the superiority of sexual reproduction [23].

An ideal method to mitigate overfitting and in the meantime improve model performance is to combine multiple models, which are different from each other while all perform well in some sense. To be different, these models could have different architecture or be trained on different data sets. However, it is intractable and computationally hard to achieve good performance for all individual models in practice. On the other hand, this idea needs a large amount of data.

Dropout approximates the combination of multiple models while solving the above-mentioned problems. The main idea of dropout is to randomly remove a part of neurons along with their connections during training, as shown in figure A.2. The crossed neurons are dropped, thus, during forward propagation, these dropped neurons are not engaged, and the weights on the edges that attach to these neurons are not updated during back-propagation.

Applying dropout to the standard network forms a 'thinned' network that contains only the neurons retained. During the training process that contains a lot of iterations, at each iteration, a 'thinned' network is sampled and the weights of retained connection are adapted based on BP. A network that has  $n$  neurons has  $2^n$  possible 'thinned' networks and they all share weights. Thus, dropout allows getting exponentially many different 'thinned' networks.

As discussed above, combining different networks alleviates overfitting and improves performance. At prediction time, instead of averaging predictions from all 'thinned' networks, a single network that contains all neurons with scaled-down

weights could approximate the 'combination' idea. For instance, if the dropout ratio during the training process is  $p$ , then the weights of the test network are multiplied by  $p$ . In this way, only one network, whose outputs from each neuron are the same as the average value of all 'thinned' networks, is involved in prediction computation.

Regarding the regularization effect, dropout reduces "co-adapting". The so-called "learning" of neural networks refers to making the weights meet the desired characteristics. Each neuron depends on its neighbors, and over-reliance could cause overfitting. Letting a random part of neurons taken away each time, the remaining neurons have to supplement the function of disappearing neurons, so that the "co-adapting" is mitigated. And the whole network becomes a collection of many independent networks and each network is a solution to the same problem. Thus, the benefit of dropout is that the network is less sensitive to changes in the weights, increasing the generalization ability and avoiding overfitting.

Gal et al. have shown that predictive distribution could be obtained from dropout-based neural networks [7]. Different from the prediction process introduced above, where dropout is prohibited and weights are scaled-down, they proposed to keep the dropout functionality during prediction and calculate the predictive distribution as equation A.5, which is a weighted integral over all possible weights.

$$q(y|x) = \int p(y|x, \mathbf{w}) \times q(\mathbf{w}) d\mathbf{w} \quad (\text{A.5})$$

In practice, the integration above is approximated by Monte-Carlo (MC) sampling. Simply put, each input passes through the network multiple times with random neurons dropped for each time, then, the prediction mean, variance, and distribution are calculated in some way, such as empirically averaged, from model outputs. This dropout-based approach is also referred to as MC-Dropout.

## Deep Ensemble

As introduced above, BNN is a statistically well-defined approach to quantifying prediction uncertainty, but it is computationally expensive. Lakshminarayanan et al. proposed deep ensemble as an alternative to BNN, which is simpler and more feasible [10]. Compared to MC-Dropout which approximates model ensemble, this method is an explicit ensemble algorithm.

Ensemble means training multiple base learners and combining them. For regression tasks, standard neural networks that have two outputs are chosen as base learners, aiming to obtain predictive distribution. Different from point prediction neural networks which only provide a single value for each predicting variable, the base learners output two values in the last layer, corresponding to mean  $\mu$  and

variance  $\sigma$  of predictive distribution, and the predictive distribution is assumed to be a Gaussian constructed by  $\mu$  and  $\sigma$ . In other words, the last layer is probabilistic. Thus, the training criterion, i.e., loss function is negative log-likelihood  $-\log p(y|Gaussian(f(x, w)))$ , where  $y$  is actual output and  $f(x, w)$  is model prediction given input  $x$ .

Given the above structure, each base learner is initialized with various random parameters (weight) values. Randomization-based ensemble approach, which means base learners are trained parallel and do not interact, is adopted. Unlike conventional ensemble algorithm where base learners are trained on different data set to make them different, for deep ensemble all base networks are trained on the entire training data set, since various initialization is enough to produce different networks.

After training, the uniformly-weighted mixture models are obtained. For regression problems, the average of Gaussian distributions is calculated for each output. In conclusion, deep ensemble produces multiple different networks, and each network has a probabilistic layer. After being predicted by all networks, the mixture distribution is extracted.