University of Stuttgart
Institute of Industrial Automation
and Software Engineering

Deep Learning-based Monitoring of Urban Traffic using MOBATSim

Presented by: Nikita Jhawar

Supervised by: Sheng Ding

Examined by: Jun.-Prof. Dr.-Ing. Andrey Morozov

THESIS WORKFLOW

1. Adapting MOBATSim Model and Fault Injection
2. Faulty and Fault Free Data Collection
3. Training DL Models for Fault Classification
4. Result & Comparison between DL models & Ensemble

TABLE OF CONTENTS

MOBATSim

MOTIVATION

1

The automotive sector needs new tools and techniques to assess and analyze the safety of the vehicles as well as the traffic.

2

Automated Traffic system helps in optimizing transport system and make it safer.

3

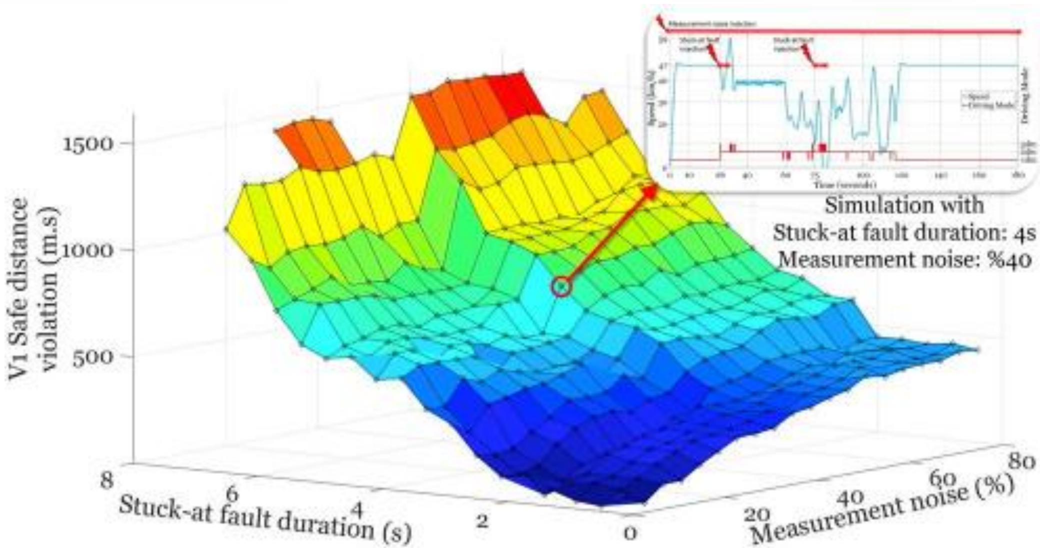
MOBATSim allows simulation-based fault injection in order to evaluate the safety of autonomous driving systems [1].

4

Deep Learning-based anomaly detection in Cyber-Physical Systems have proved to be very effective [2].

Related Work

The use of MOBATSim for fault error failure chain analysis has been presented using an illustrative case study showing the effects of fault on the overall traffic safety in MOBATSim [3] by injecting fault in one vehicle in the traffic model.

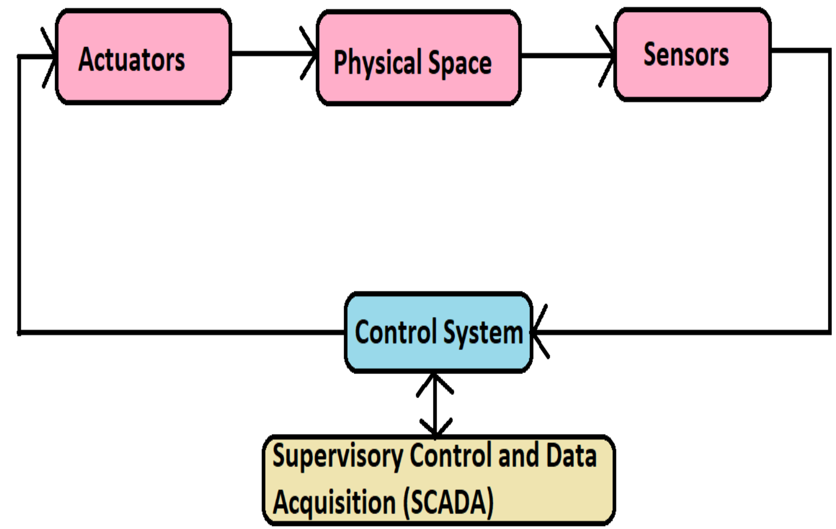


Safe distance violation as a function of stuck-at-fault duration and sensor noise[3].

INTRODUCTION & FAULT INJECTION

Cyber-Physical System

- Cyber physical systems (CPS) is defined as systems of collaborating computational entities that are in thorough connection with the neighboring physical world as well as its ongoing processes, thus enabling data processing services [4].
- MOBATSim is a type of CPS, allowing the user to develop automated driving algorithms



Generic CPS Diagram [4]

MOBATSim: Model-based Autonomous Traffic Simulation Framework

.The MOBATSim traffic Model has 10 vehicles sharing their speed, rotation, and translation data after simulation

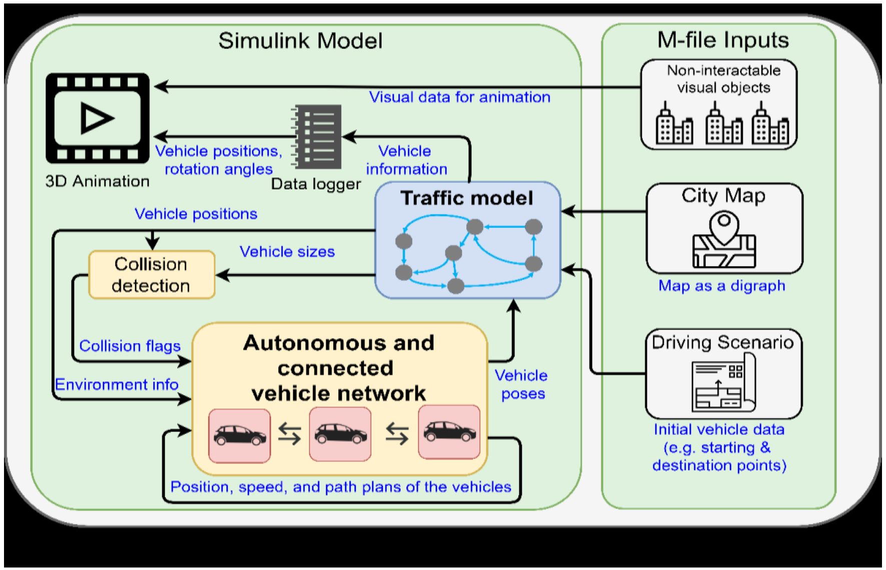
FI block is used to inject fault in 2 vehicles at speed and distance sensor

Three types of sensor faults are injected namely Noise, StuckAt and Offset

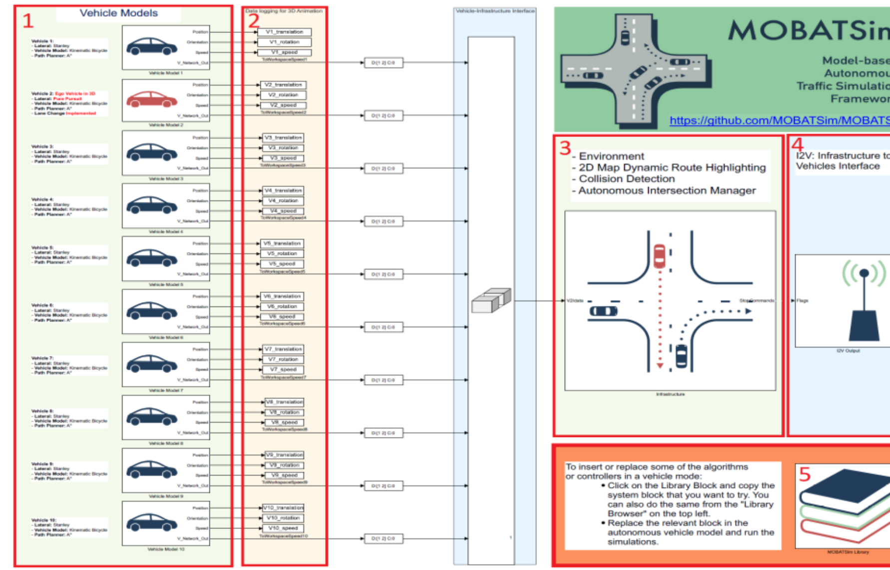
Simulation is run for 3 different driving scenarios with different fault values

Two types of data i.e., Univariate and Multivariate data is collected

Description Of Simulink Model

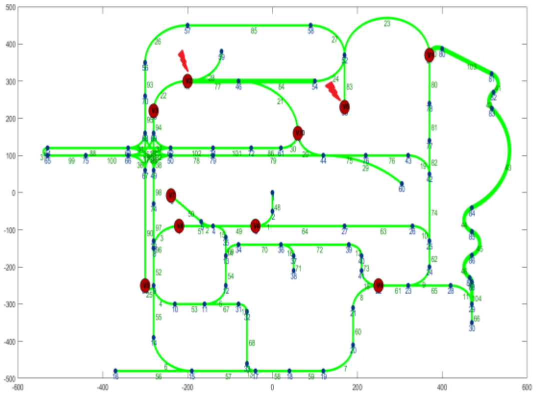


Workflow Diagram [1]

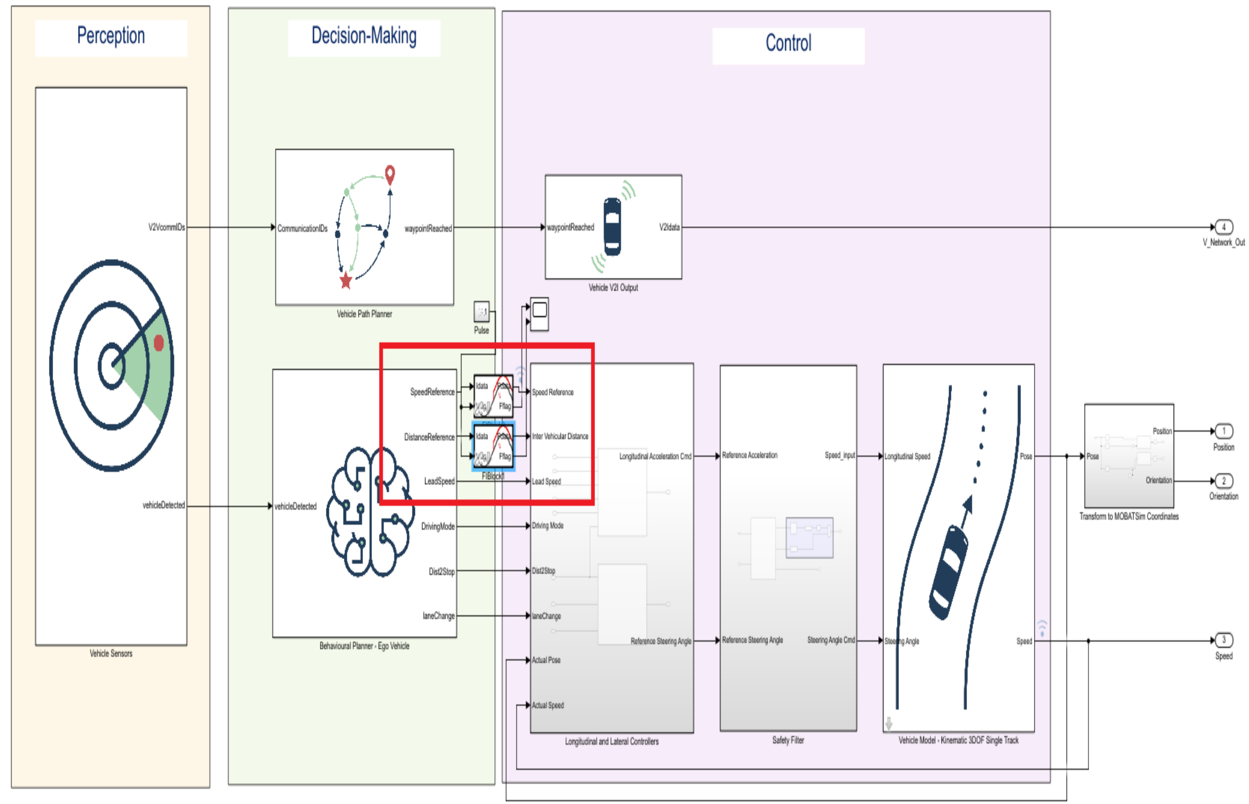


Main Simulink Model [1]

Vehicle Model After Fault Injection

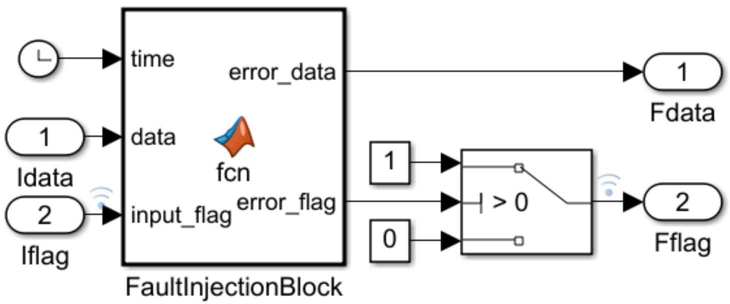


2D and 3D map plot[1]

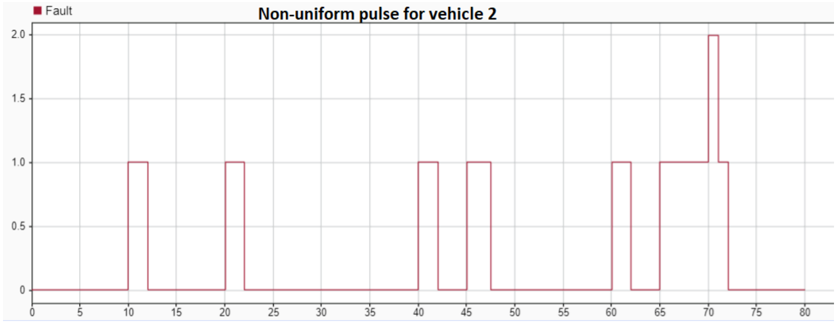


Vehicle Model after Fault Injection[1]

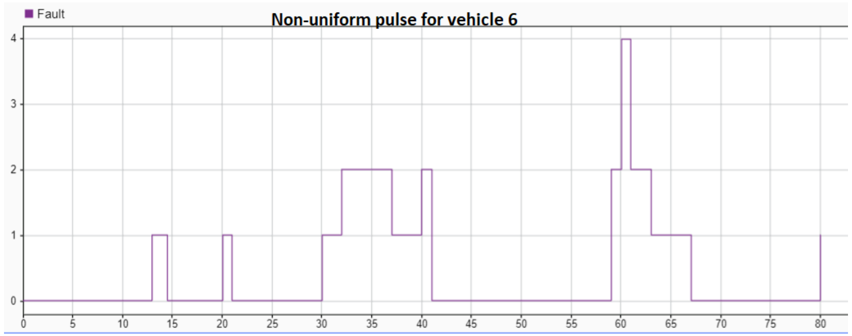
FI Block and Non-uniform Pulse Generator



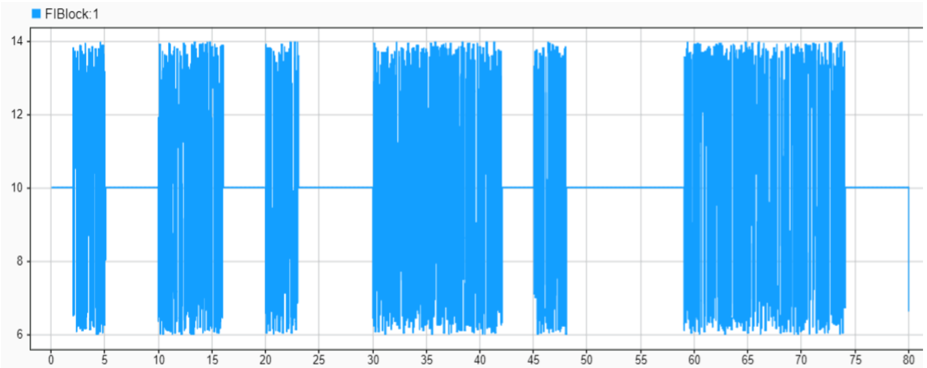
FI Block[11]



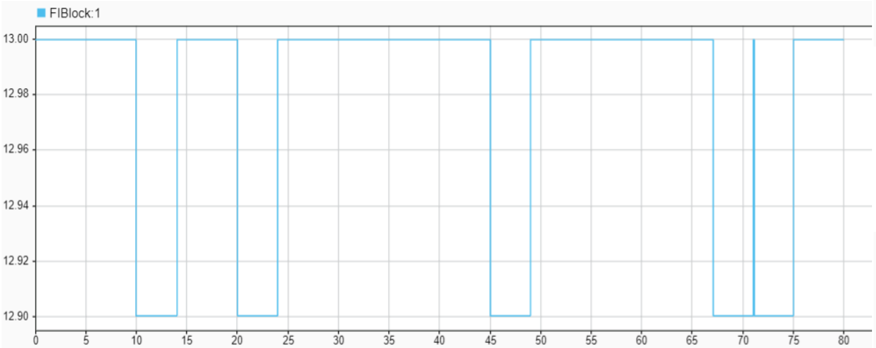
Fault Category	Fault Value	Fault Delay	Fault Duration
Noise	20%, 30%, 40%, 50%	1, 2, 3, 4, 5	1, 2, 3, 4, 5
Stuck-at	Stuck to last value	1, 3, 5, 7, 9, 11, 13, 15	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
Bias/Offset	-1, -2, -3	1, 2, 3, 4, 5	2, 3, 4, 5, 6, 7



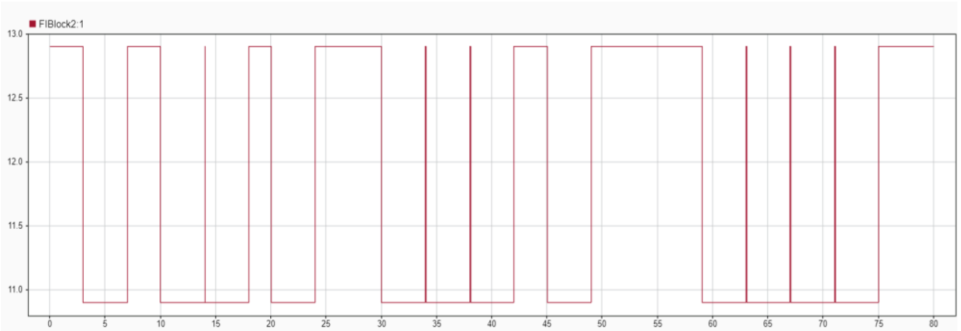
Three Categories of Fault



Noise Fault : Sampling Rate 0.02sec



Stuck-at Fault : Sampling Rate 0.02sec

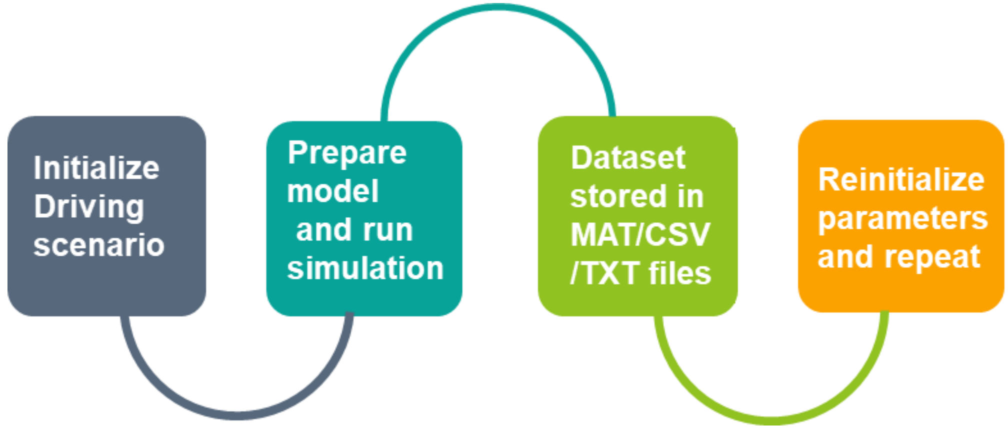


Offset Fault : Sampling Rate 0.02sec

DATA COLLECTION

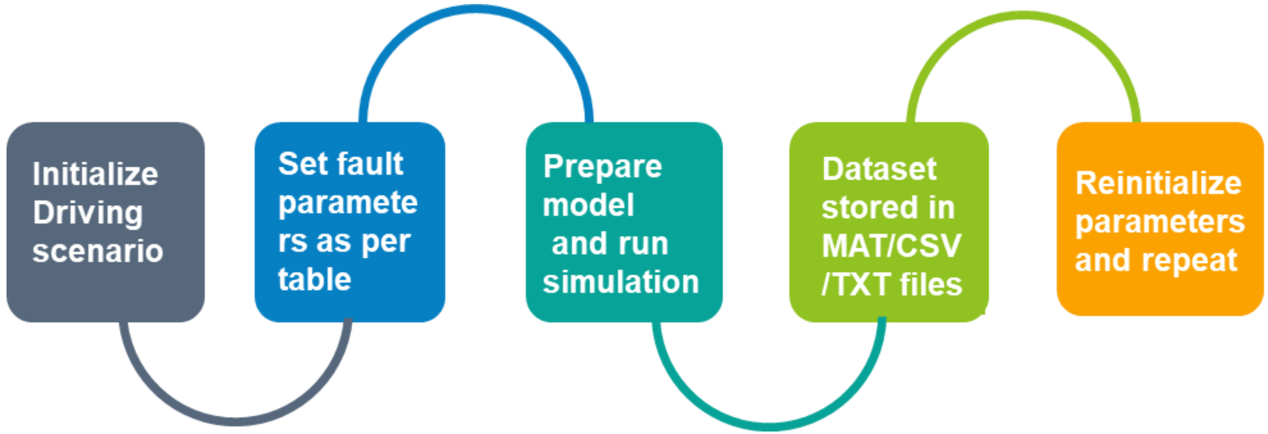
Fault-Free Data Collection

- The process is repeated for three driving scenarios:
  - Platoon Control
  - Road Merge Collision
  - Urban City Traffic



- The duration of the simulation is 80 seconds, and the data is re-sampled from 0.005 seconds to 0.02 seconds for data compression. Therefore, each data set generated for one simulation of 80 secs has 3999 data points (entries/rows)

Faulty Data Collection



- Data is collected for random fault delays and fault duration chosen from the table in previous slides for Noise, Stuck-at, and Offset/Bias for both vehicle 2 and vehicle 6 for all three scenarios.
- Data is generated for each univariate and multivariate data. Around more than 150 faulty data files are collected with each data set generated for one simulation of 80 secs having 3999 data points

CSV Data Format

Time	Speed	Label
0 to 80 seconds	0 to maximum Speed (Dependent on the scenario)	0,1,2 or 3
Time interval of 0.02 secs	Maximum speed defined in simulation files	Depending on fault category

Univariate Data

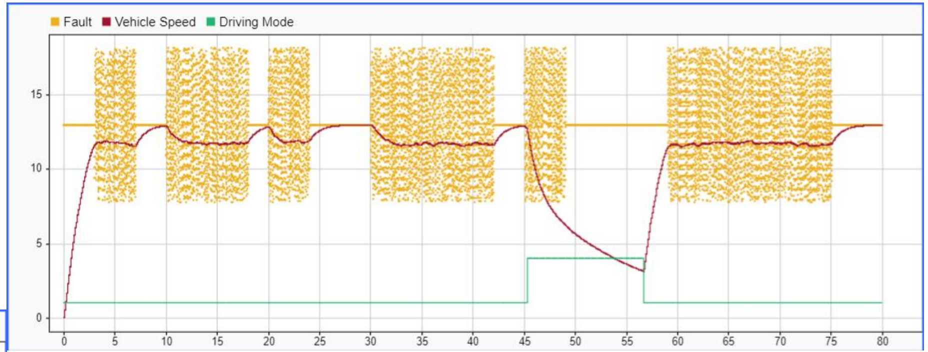
Dimension of Final Dataset:
287928 rows × (3 or 6) columns
72 Datasets combined having 3999 samples each

Anomaly Percentage:
Fault Free: 53.8%
Noise: 17.5%
Stuck-At: 13.75%
Offset: 14.7%

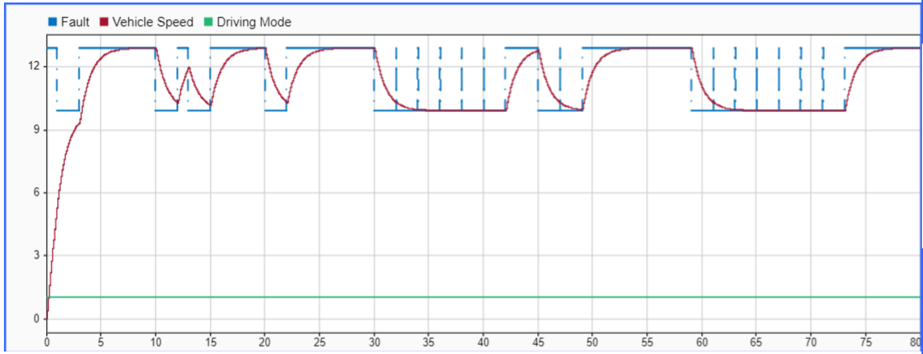
Time	Speed	Rotation	Position	Translation	Label
0 to 80 seconds	0 to maximum Speed (Dependent on the scenario)	-3 to 3	-400 to 400	-500 to 300	0,1,2 or 3
Time Interval of 0.02 seconds	Maximum speed defined in simulation files	Values pre-defined	Values pre-defined	Values pre-defined	Depending on fault category

Multivariate Data

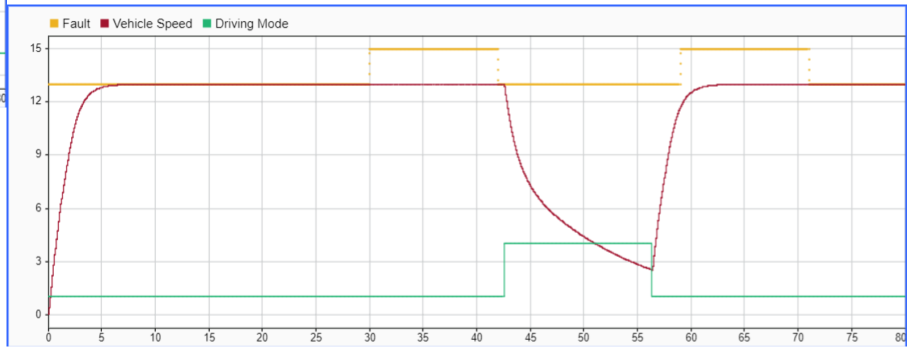
Sample Faulty Data for Vehicle 2



Noise fault injection

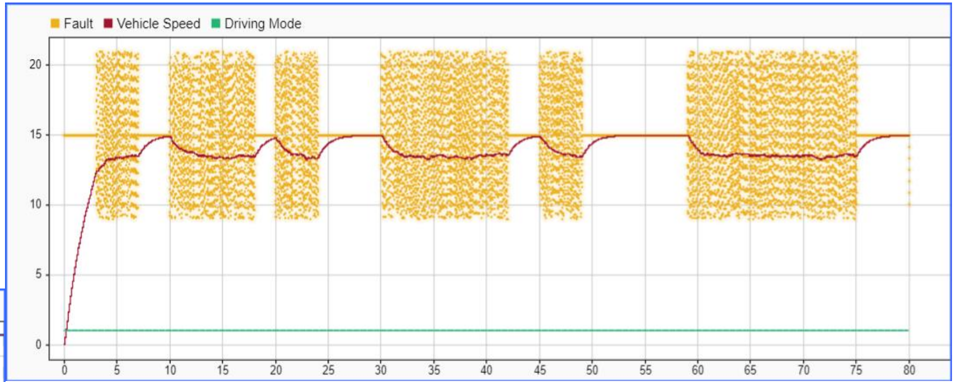


Offset fault injection

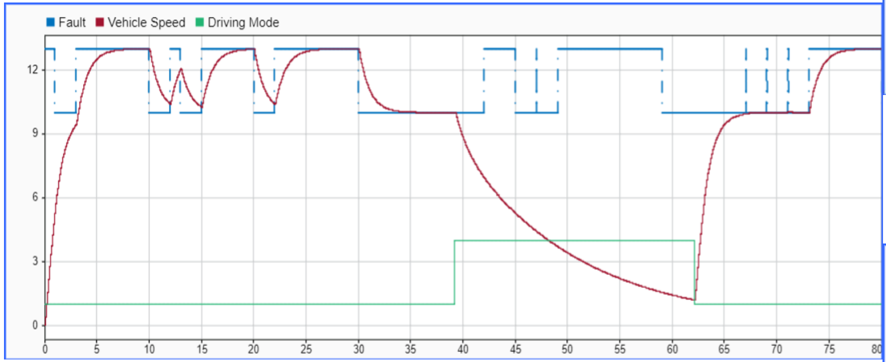


Stuck-At fault injection

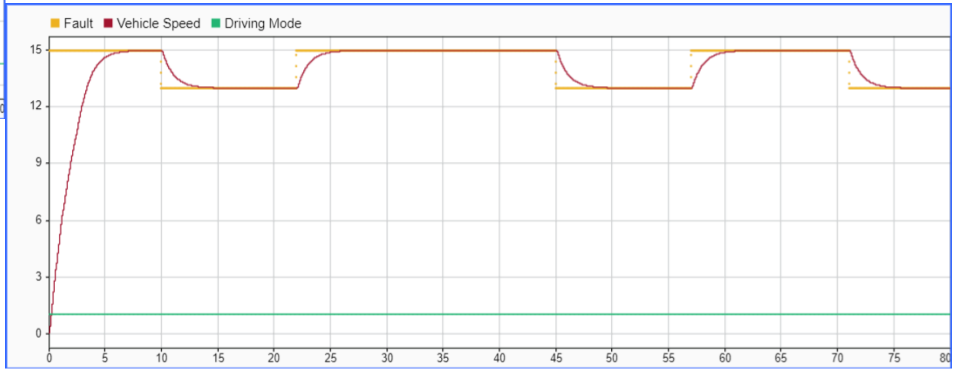
Sample Faulty Data for Vehicle 6



Noise fault injection

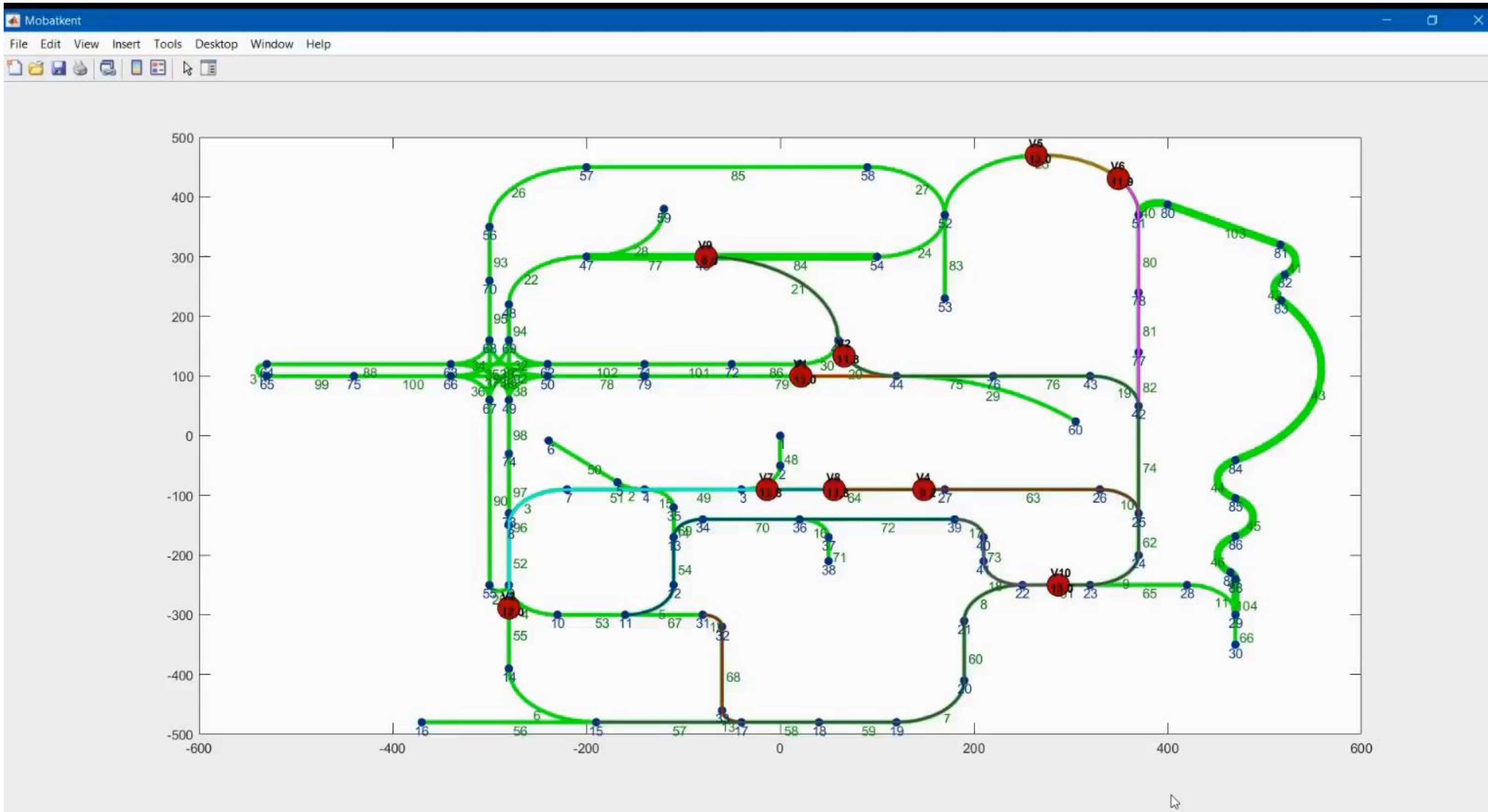


Offset fault injection



StuckAt fault injection

Demonstration Video



FAULT DETECTION USING DL MODELS

Steps for Data Pre-Processing

1 Ground Truth Labeling

0 denote fault-free scenario, 1 for noise fault, 2 for stuck-at and 3 for offset

2 Data Shuffling

In order to build a better DL model, we shuffle the data set for our further use

3 Train-Test Data Split

The data is then split randomly with 0.33 part being test data and 0.67 being the train data

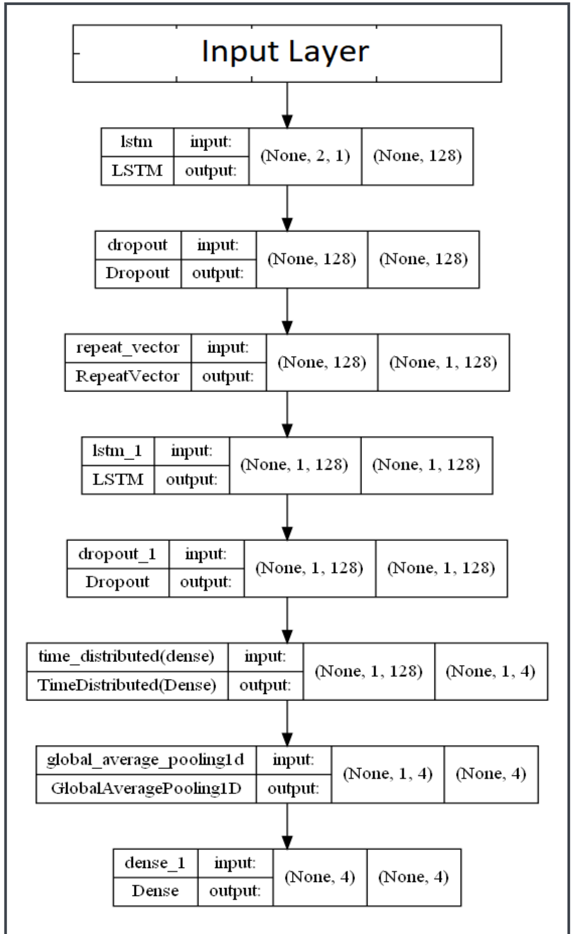
4 Categorical Conversion

Conversion of class vector to the binary class matrix in order to use with categorical cross-entropy

Deep Learning Models Implemented

- Recurrent Neural Network [5] consists of three types i.e., Simple RNN, Gated Recurrent Unit (GRU) and Long Short-Term Memory(LSTM)
- Autoencoder is a type of neural network that is referred to learn a compressed representation of raw data as it is composed of an encoder and a decoder sub-models
- Transformer models apply an evolving set of mathematical techniques, called attention or self-attention, to detect subtle ways even distant data elements in a series influence and depend on each other
- In this thesis, we train and test our data with 6 different DL models i.e., Simple RNN, GRU-Simple RNN, BiGRU-BiSimple RNN [6], Autoencoders [9], Transformers [8], and hybrid LSTM-Autoencoders [7]
- The three main classes of ensemble learning methods are **bagging**, **stacking**, and **boosting** and In the end, the Stacking Ensemble Algorithm [10] is used which involves taking the outputs of sub-models as input and attempts to learn how to best combine the input predictions to make a better output prediction.

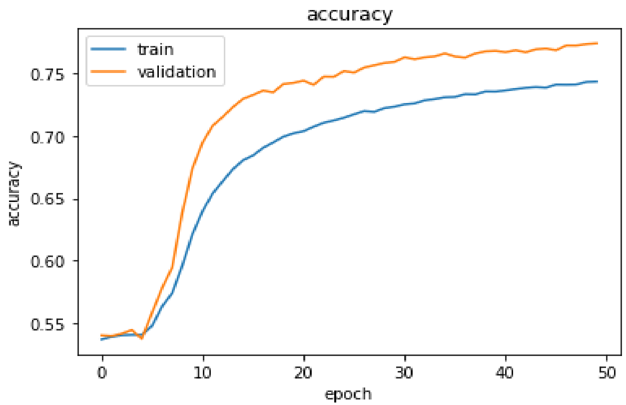
LSTM-Autoencoder Architecture

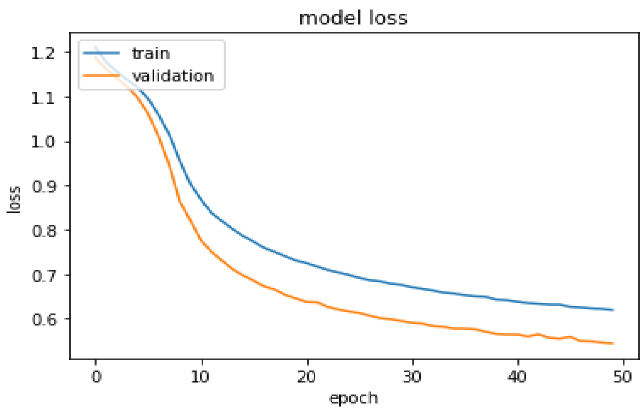


Hyper-Parameters	Value
Optimizer	Adam Optimizer
Learning Rate	0.001
Type	Sequential
Loss	Categorical Crossentropy
Batch Size	2048
No. of Epochs	50
Trainable Params	198,680
Computational Training Time (Univariate)	21 seconds /epoch
Computational Training Time (Multivariate)	36 seconds /epoch

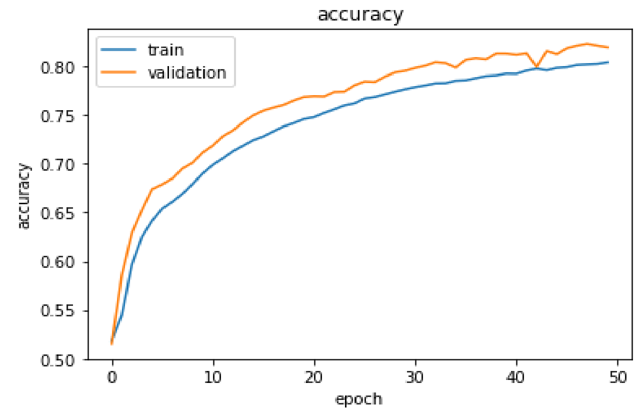
LSTM-Autoencoder Performance

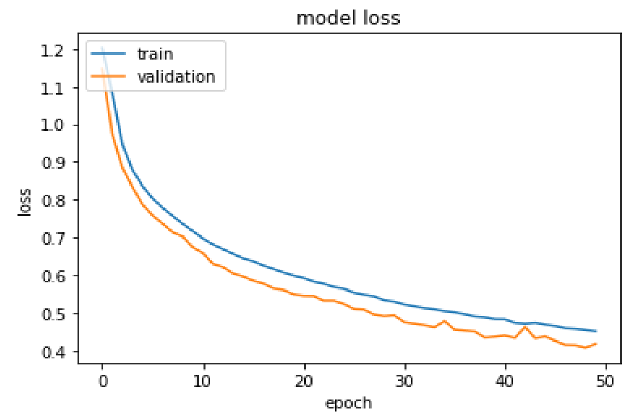
Accuracy and Loss for Uni-variate Data





Accuracy and Loss for Multivariate Data



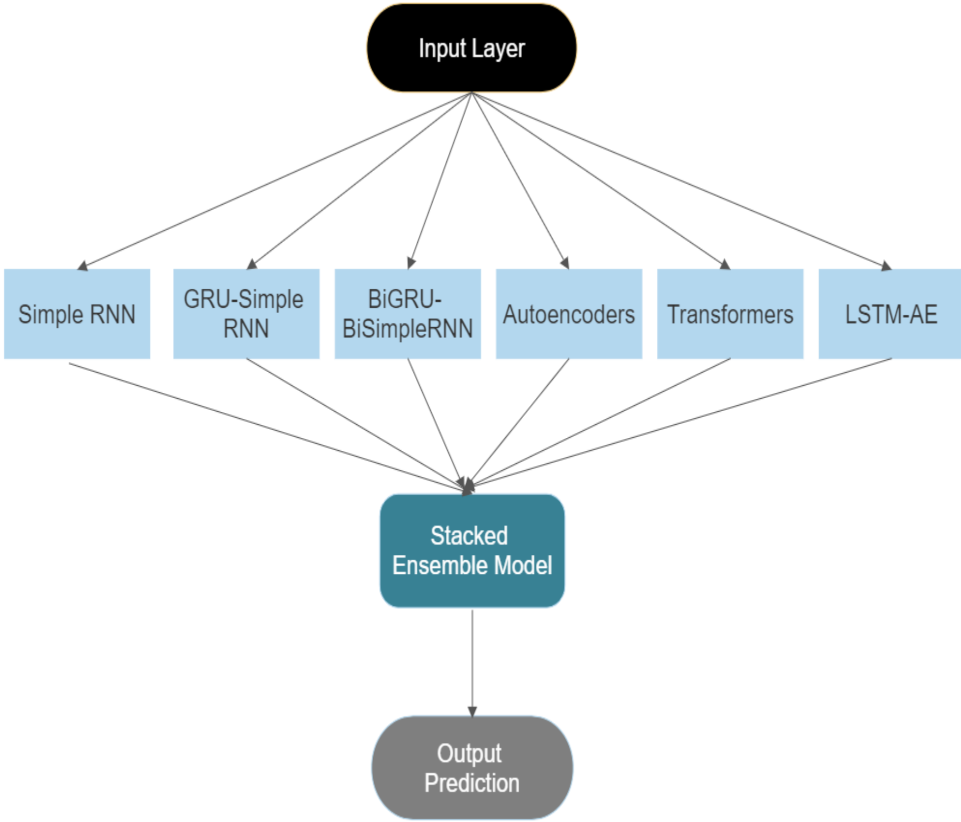


Stacking Ensemble Algorithm

- Stacking procedure consists of two levels:

1. **Level 0**: The level 0 data is the training dataset inputs and level 0 models learn to make predictions from this data.
2. **Level 1**: The level 1 data takes the output of the level 0 models as input and the single level 1 model, or meta-learner, learns to make predictions from this data

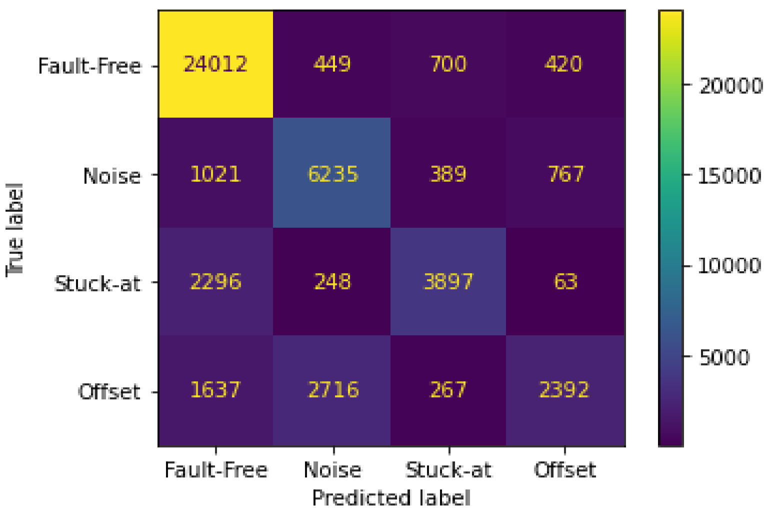
- A stacked generalization ensemble can use the set of predictions as a context and conditionally decide to weigh the input predictions differently, potentially resulting in better performance



RESULTS

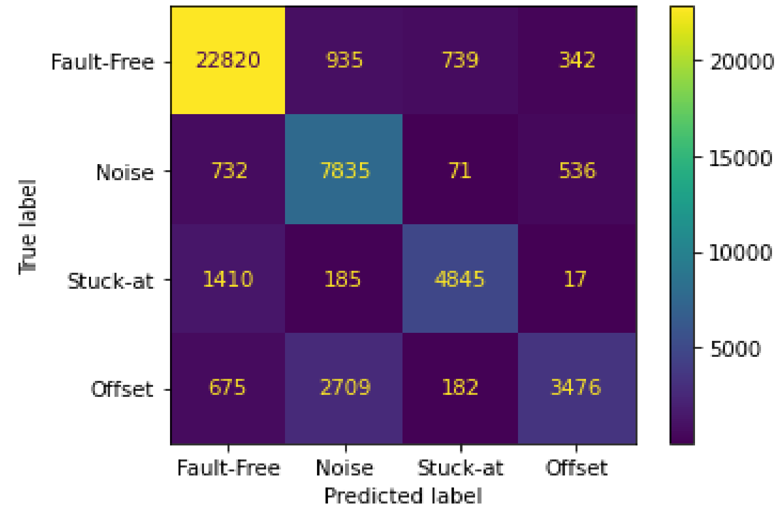
Fault Classification of LSTM-Autoencoders on Test Data

Univariate Data



	precision	recall	f1-score	support
Fault-Free	0.83	0.94	0.88	25581
Noise	0.65	0.74	0.69	8412
Stuck-at	0.74	0.60	0.66	6504
Offset	0.66	0.34	0.45	7012
accuracy			0.77	47509
macro avg	0.72	0.66	0.67	47509
weighted avg	0.76	0.77	0.75	47509

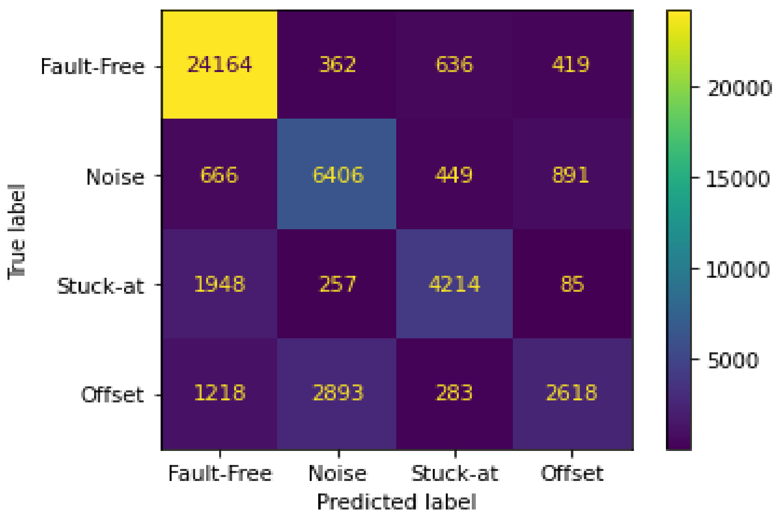
Multivariate Data



	precision	recall	f1-score	support
Fault-Free	0.89	0.92	0.90	24836
Noise	0.67	0.85	0.75	9174
Stuck-at	0.83	0.75	0.79	6457
Offset	0.80	0.49	0.61	7042
accuracy			0.82	47509
macro avg	0.80	0.75	0.76	47509
weighted avg	0.83	0.82	0.82	47509

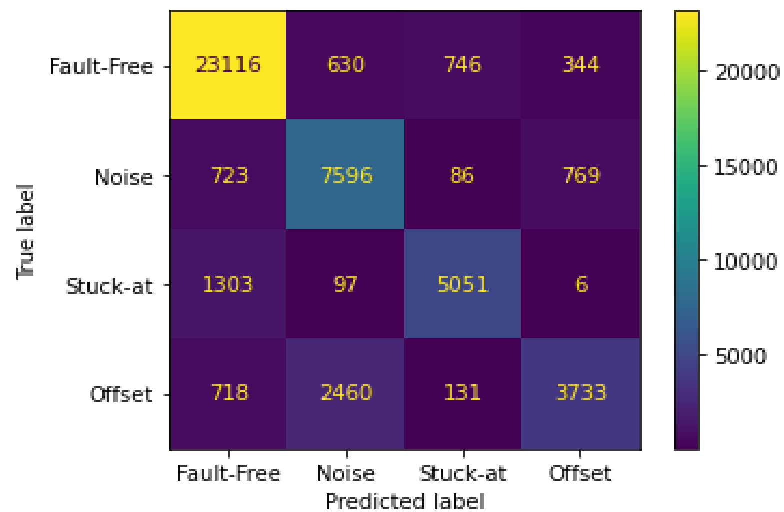
Fault Classification of Stacked Ensemble Algorithm

Univariate Data



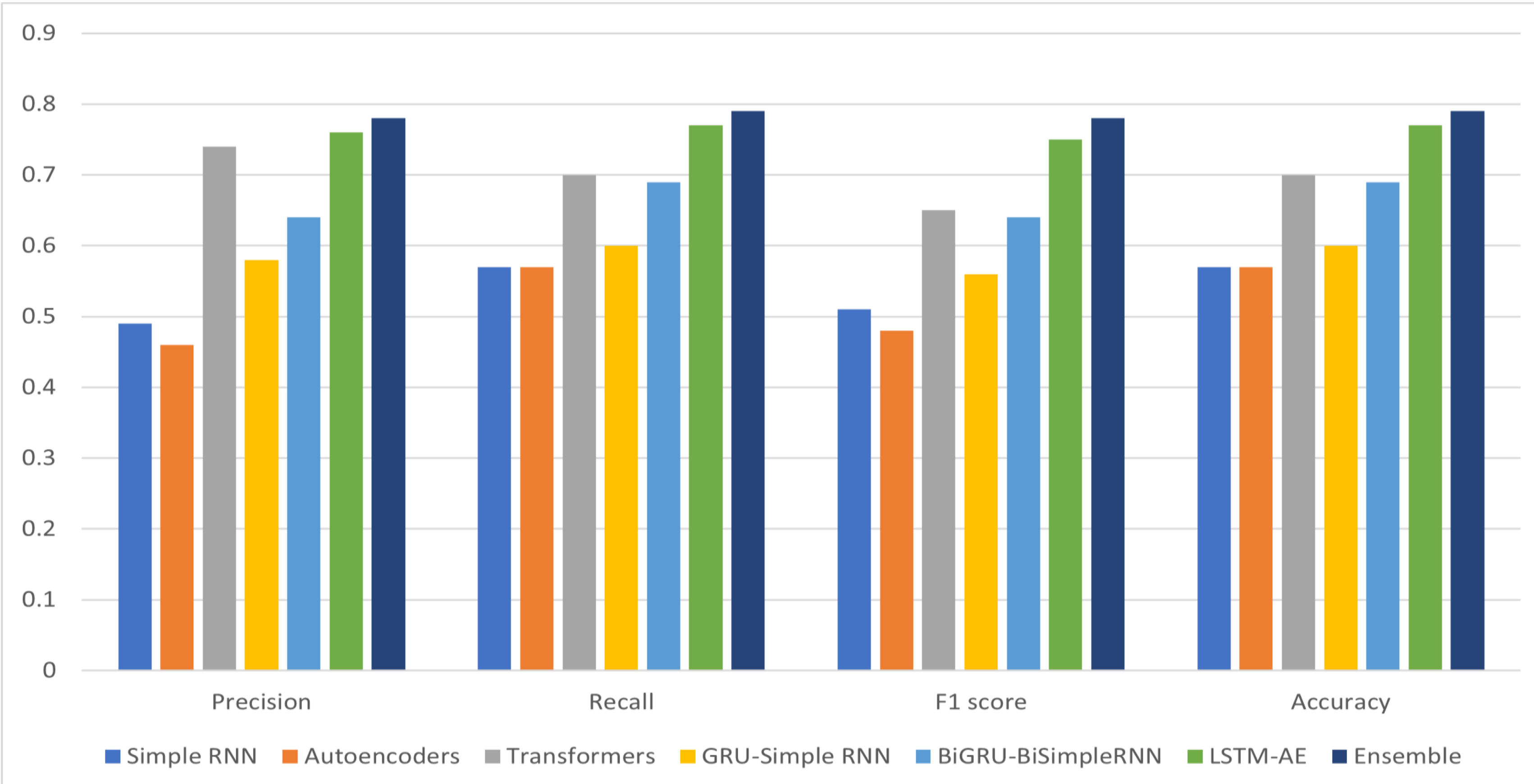
	precision	recall	f1-score	support
Fault-Free	0.86	0.94	0.90	25581
Noise	0.65	0.76	0.70	8412
Stuck-at	0.75	0.65	0.70	6504
Offset	0.65	0.37	0.47	7012
accuracy			0.79	47509
macro avg	0.73	0.68	0.69	47509
weighted avg	0.78	0.79	0.78	47509

Multivariate Data

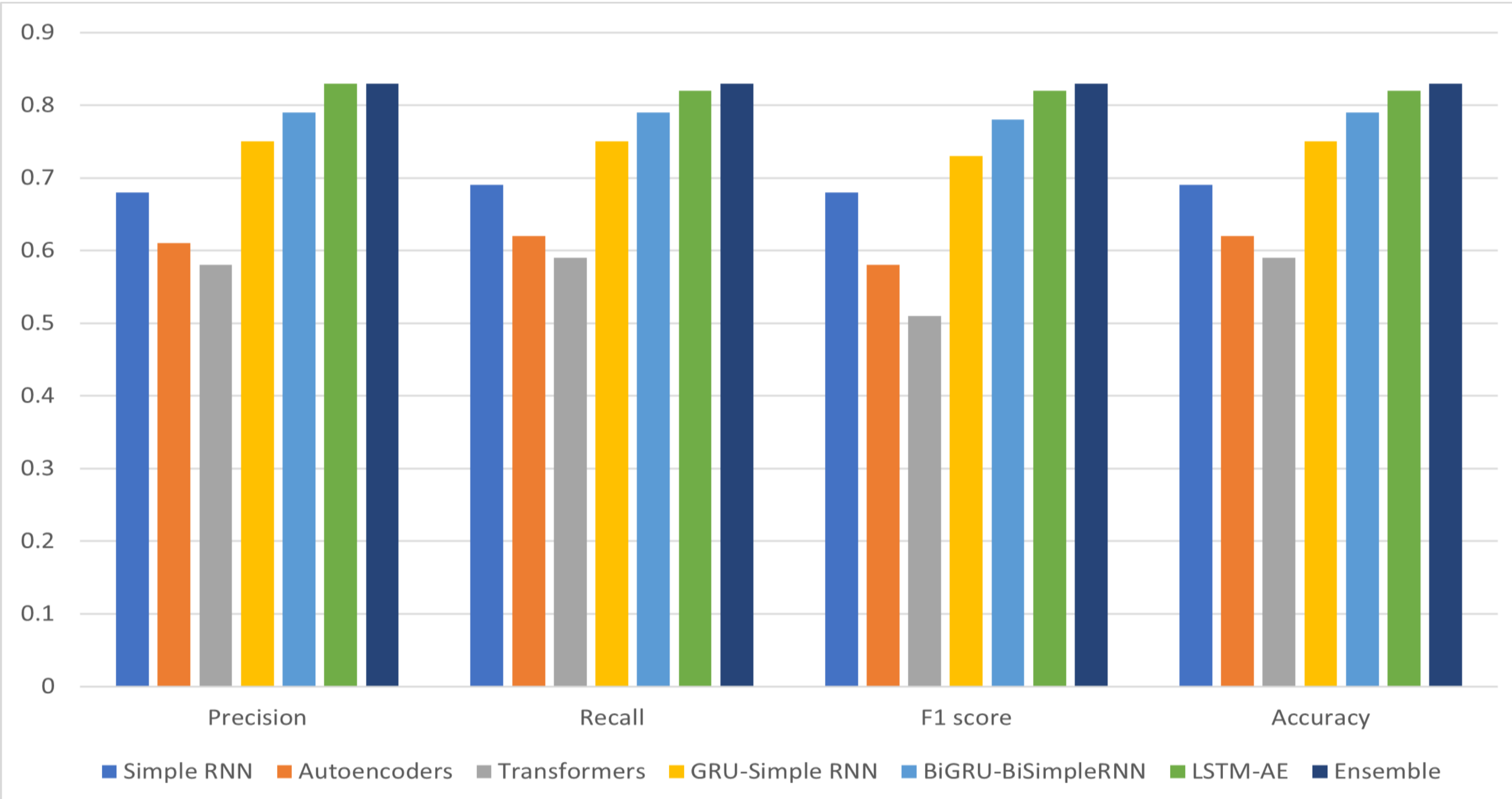


	precision	recall	f1-score	support
Fault-Free	0.89	0.93	0.91	24836
Noise	0.70	0.83	0.76	9174
Stuck-at	0.84	0.78	0.81	6457
Offset	0.77	0.53	0.63	7042
accuracy			0.83	47509
macro avg	0.80	0.77	0.78	47509
weighted avg	0.83	0.83	0.83	47509

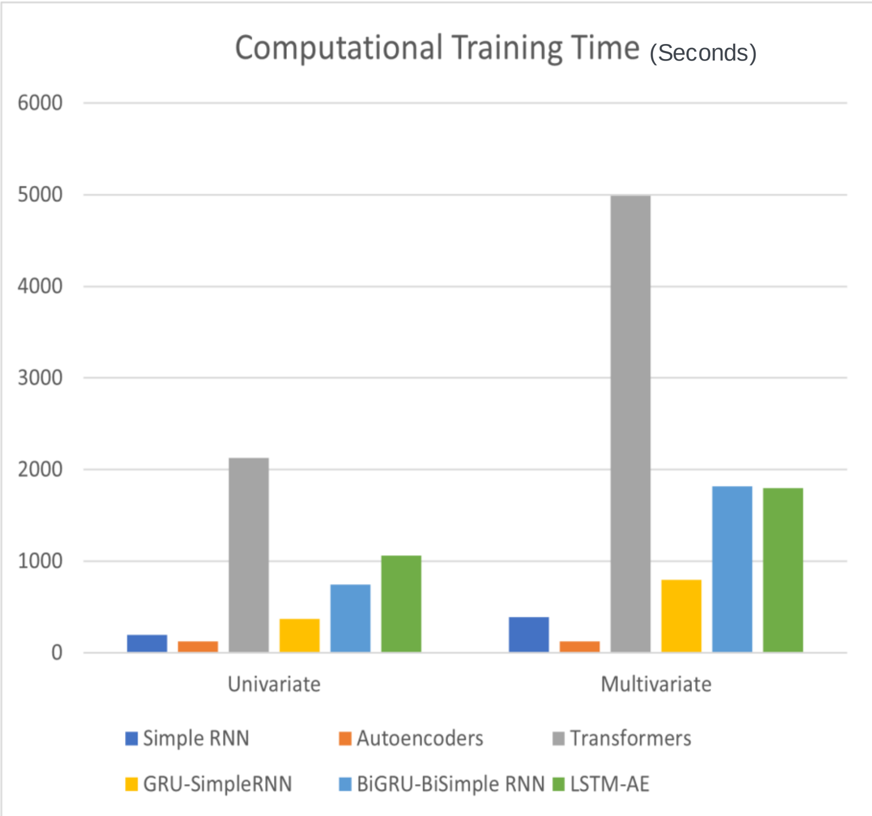
Comparison between DL Models for Univariate Data

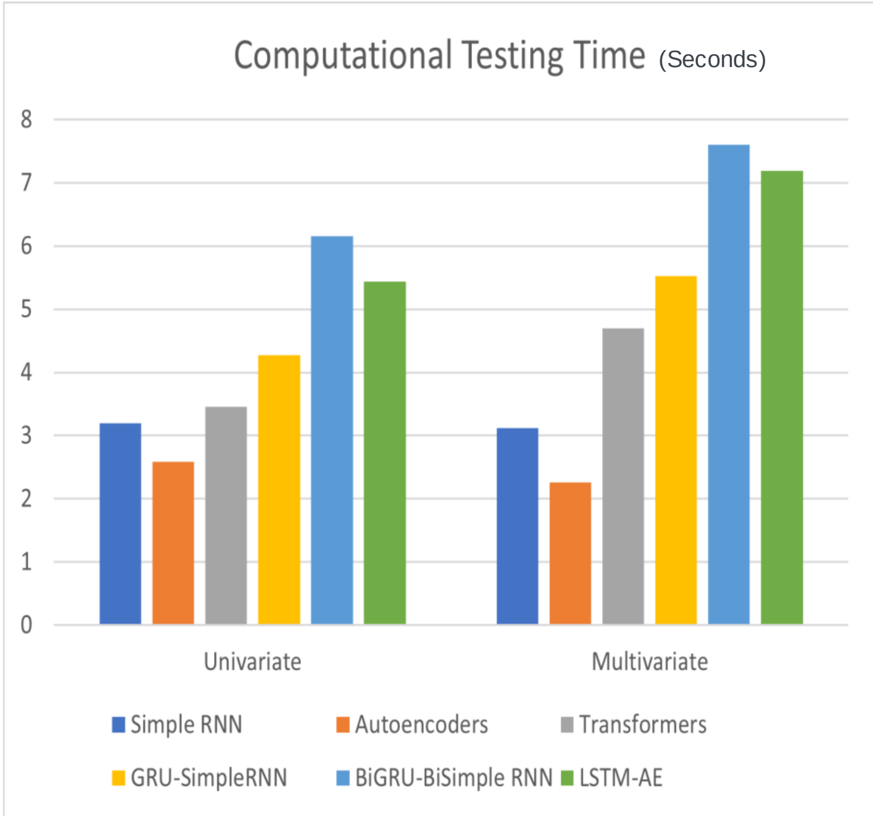


Comparison between DL Models for Multivariate Data



Representation of Computational Time for DL Models





CONCLUSION & FUTURE SCOPE

Conclusion

01 The Traffic Model of MOBATSim consisting of 10 vehicles with 3 different scenarios is adapted. Sensor Faults are injected using FI block at speed and distance sensor of Vehicle 2 and vehicle 6 .

02 Faulty and Fault-Free (univariate and multivariate) data is generated and eventually, labeled and pre-processed

03 Simple RNN, GRU-Simple RNN, Bidirectional GRU-Bidirectional Simple RNN, Autoencoders, Transformers, and LSTM-Autoencoder architectures are used for training and testing both types of data and later the stacking Ensemble Technique is also used in order to improve accuracy.

04 LSTM-Autoencoders perform the best out of all the DL models used for both types of data and Ensemble improves the accuracy by increasing from 77% to 79% with univariate data and from 82% to 83% with multivariate data.

Future Scope

1

Data Collection with different changes in the Vehicle Model for MOBATSim

2

Other fault types like hardware and network faults can be implemented

3

Other DL, ML, and Ensemble algorithms can be used for fault classification.

4

Different pre-processing methods can be used for other architectures

REFERENCES

[1] https://mobatsim.com/

[2] Deep Learning-Based Anomaly Detection in Cyber-Physical Systems: Progress and Opportunities, Yuan Luo, Ya Xiao, Long Cheng, Guojun Peng, Danfeng Daphne Yao, arXiv:2003.13213v2 [cs.CR]

[3] Saraoglu, M., Morozov, A., & Janschek, K. (2019). MOBATSim: MOdel-Based Autonomous Traffic Simulation Framework for Fault-Error-Failure Chain Analysis. IFAC-PapersOnLine, 52(8), 239–244. Elsevier BV. Retrieved from https://doi.org/10.1016%2Fj.ifacol.2019.08.077[4] Monostori L. (2018) Cyber-Physical Systems. In: The International Academy for Production, Chatti S., Tolio T. (eds) CIRP Encyclopedia of Production Engineering. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-35950-7_16790-1

[5] A. Nanduri and L. Sherry, "Anomaly detection in aircraft data using Recurrent Neural Networks (RNN)," *2016 Integrated Communications Navigation and Surveillance (ICNS)*, 2016, pp. 5C2-1-5C2-8, doi: 10.1109/ICNSURV.2016.7486356.

[6] https://medium.com/@felixs_76053/bidirectional-gru-for-text-classification-by-relevance-to-sdg-3-indicators-2e5fd99cc341

[7] “LSTM-Autoencoder based Anomaly Detection for Indoor Air Quality Time Series Data”, Yuanyuan Wei, Julian Jang-Jaccard, Wen Xu, Fariza Sabrina, Seyit Camtepe, Mikael Boulic, arXiv:2204.06701v1 [cs.LG]

[8] “TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data” , Shreshth Tuli, Giuliano Casale, Nicholas R. Jennings, arXiv:2201.07284v6 [cs.LG]

[9] https://towardsdatascience.com/anomaly-detection-using-autoencoders-5b032178a1ea

[10] https://machinelearningmastery.com/stacking-ensemble-for-deep-learning-neural-networks/

[11] https://de.mathworks.com/matlabcentral/fileexchange/75539-fault-injection-block-fiblock

THANK YOU