Infrastructure-based End-to-End Learning and Prevention of Driver Failure

Noam Buckman* MIT CSAIL nbuckman@mit.edu

Shiva Sreeram* Caltech sasreera@caltech.edu

Mathias Lechner MIT CSAIL mlechner@mit.edu

Yutong Ban MIT CSAIL yban@mit.edu

Ramin Hasani MIT CSAIL rhasani@mit.edu

Sertac Karaman MIT LIDS sertac@mit.edu

Daniela Rus MIT CSAIL rus@mit.edu

Abstract

Intelligent intersection managers can improve safety by detecting dangerous drivers or failure modes in autonomous vehicles, warning oncoming vehicles as they approach an intersection. In this work, we present FailureNet, a recurrent neural network trained end-to-end on trajectories of both nominal and reckless drivers in a scaled miniature city. FailureNet observes the poses of vehicles as they approach an intersection and detects whether a failure is present in the autonomy stack, warning cross-traffic of potentially dangerous drivers. FailureNet can accurately identify control failures, upstream perception errors, and speeding drivers, distinguishing them from nominal driving. The network is trained and deployed with autonomous vehicles in a scaled miniature city. Compared to speed or frequency-based predictors, FailureNet's recurrent neural network structure provides improved predictive power, yielding upwards of 84% accuracy when deployed on hardware.

1 Introduction

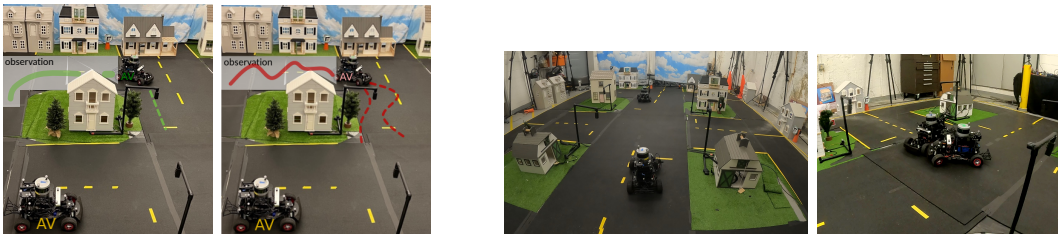
Safety is critical for the adoption of autonomous vehicles (AVs) on roads, especially as an increasing number of vehicles are deployed on the road. Given that failures and errors will always exist, methods must be developed for identifying issues with autonomous vehicles and alerting vehicles with enough time to take action. Infrastructure-based methods, such as intelligent intersection managers, can observe drivers for longer duration for improved failure detection. In addition, control of the intersection provides an extra level of safety, especially for cross-traffic collisions.

In this work, we consider an intelligent traffic light that monitors vehicles for failures and warns oncoming traffic to prevent collisions. Existing approaches such as driver monitoring systems require in-cabin sensor placement for driver monitoring which can capture more information but requires access to the vehicle itself, whereas an external monitor does not require access to the vehicle. Furthermore, current approaches are typically limited to vehicles observed within the field-of-view of the vehicle, whereas external monitoring from an intersection manager can monitor vehicles as they approach an intersection. For a more detailed related works, see Appendix A.

In our approach, an intersection manager observes a vehicle's trajectory as it drives near an intersection and uses FailureNet, a recurrent neural network (RNN), to detect whether a driver's behavior is caused by a planning or actuator failure (Fig. 1). Our learning-based approach is trained to detect failures from generated data within a 1/10th-scaled miniature city testbed where multiple autonomous vehicles are deployed simultaneously. We induce vehicle failures in the scaled hardware, ranging

*equal contribution

NeurIPS 2022 Workshop on Robot Learning: Trustworthy Robotics, Virtual, Virtual



(a) Nominal driver (b) Reckless driver (c) Hardware Testbed (d) Preventing Collisions

Figure 1: FailureNet observes nominal drivers (**a**) or reckless drivers (**b**) and warns oncoming traffic. We train and deploy on a scaled testbed (**c**) to prevent AV collisions (**d**).

from control failures (injecting noise to speed and steering) to perception failure, and train FailureNet on this novel dataset. We demonstrate the accuracy of FailureNet and our ability to warn oncoming traffic by deploying in a novel hardware testbed with multiple vehicles and compare to multiple baseline approaches.

2 Problem Statement

The goal of this work is to successfully identify vehicles with planning or sensor failures before the vehicles arrive at an intersection. Specifically, we consider an intersection and the surrounding roads flowing into the intersection that is monitored by an intelligent intersection observer and manager. We assume that under normal operation (nominal driving), each vehicle j navigates to various locations in the city autonomously, with a high-level route planning, low-level path planning, and motion control. A vehicle failure is defined as a significant degradation of one or more sub-components of the autonomous vehicle, for example, decision-making, perception, or low-level control. We assume that a vehicle's failure persists through the duration of driving and is represented by a latent failure variable, $z_j \in \{0,1\}$ where $1 = Unsafe$, $0 = Safe$.

The goal of the intersection manager is to observe the vehicles and (1) detect whether a vehicle is failing and if so, (2) mitigate intersection collisions by warning oncoming traffic. The intersection manager only has access to information observable externally. Specifically, the intersection manager observes pose of each of agent $p_{j,t} = [x_t, y_t, \theta_t]$ and the goal of traffic light is to provide an estimate $\hat{z}_{j,t}$ of whether vehicle j is experiencing a failure, and if so, communicate to incoming vehicles.

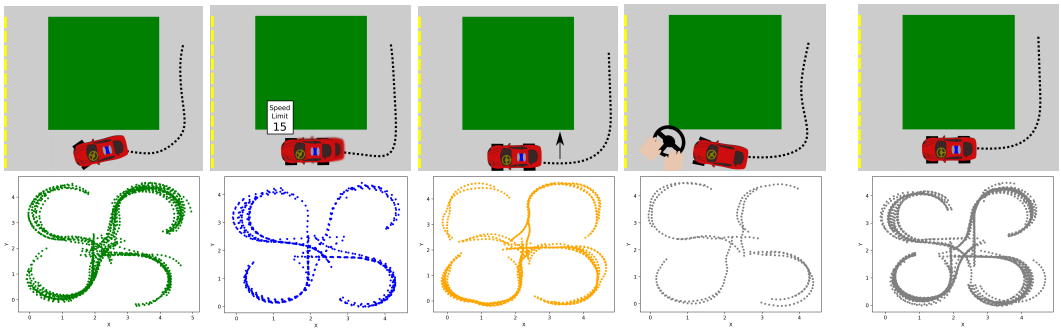
2.1 Failure Modes

On-road collisions can occur due to various types of vehicle failures. We focus on identifying failure modes that manifest in the driving behavior of the vehicle before point of collision at an intersection. We consider four types of vehicle failures motivated by reckless human driving and AV failures.

Random Periodic Control Failure The first type of failure is a random additive noise applied to the control output of the vehicle, steering, and acceleration. This failure mode is chosen to demonstrate a persistent random vehicle failure or poor driver abilities. Specifically, a random steering and speed noise is added to the desired steering and speed outputted by the autonomy stack.

Lane Detection Offset Failure Upstream failures in the perception of the vehicle, such as a failing lane detector, can lead to observable and dangerous scenarios. We consider a failing lane detector that outputs an incorrect lane line. Similar to the other failure modes, we consider a non-catastrophic failure such as a biased lateral shift of the outer lines. For each outer lane line detected, the line is shifted laterally by a distance $\bar{s}$.

Speeding Driver Speeding drivers were a contributing factor in 29% of all deaths on the road totaling 11,258 fatalities (1). We simulate a speeding driver by increasing the desired speed of the driver from 0.3m/s to 0.5m/s. The steering of the vehicle is unaffected; however, the vehicle attempts to maintain a desired speed of 0.5m/s. The high speed of the vehicle leads to increased steering oscillations due to dynamic instabilities and overshooting tight radius turns.



(a) Periodic Noise (b) Speeding Driver (c) Lane Offset (d) Manual Driver (e) Nominal Driver

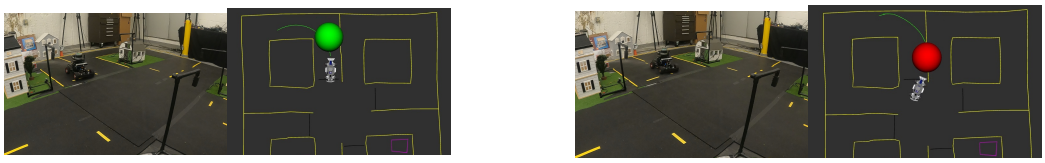
Figure 2: Failure modes deployed on the scaled cars and vehicle poses deployed in testbed.

Manual Reckless Driver Finally, to capture a wide breadth of driving styles, we consider a hybrid failure mode generate by allowing a human driver to command the vehicle in a reckless manner.

3 Training and Deploying in a 1/10th Scale Autonomous City

Hardware Testbed We utilize the MiniCity (2), a novel 1/10th scale experimentation platform, for testing and evaluating FailureNet. Scaled houses, roads, grass, and traffic lights make up a realistic aesthetic of the miniature city, with intersections and roundabouts for simulating dangerous and interactive driving scenarios. Each vehicle in the platform consists of state-of-the-art sensors, such as a Velodyne Lidar and Zed camera, and runs a full autonomy stack from high-level mission planning to low-level control. This allows us to deactivate various components of the autonomy stack to simulate catastrophic failure and measure the impact on vehicle driving. An external motion capture provides ground truth position for each vehicle and simulates GPS for onboard state estimation. Individual vehicles fuse multiple sensor modalities, including simulated GPS, to localize in the city, while we utilize the high-rate motion capture for collecting training data and evaluating the performance of FailureNet.

Training on Reckless Drivers Each miniature vehicle runs a full autonomy stack, implemented in ROS, to navigate within the city setting. Reckless driving is simulated by injecting various failure modes in the AV stack, as described in Sec. 2. For high-level failure modes such as lane detection and speeding, we modify the upstream planning nodes, whereas for low-level failures such as noisy controls, we create a noisy driver ROS node that injects random noise at the output. Each intersection contains a signalized traffic light that communicates with vehicles over ROS. Figure 1d shows an example failing AV colliding with a cross-traffic driver. During FailureNet's training, a single vehicle navigates in the city autonomously, with human monitoring and handovers in case of on or off-road collisions. Figure 2 shows the training poses captured for four failure modes: periodic noise, lane shift, speeding, and nominal driving. In addition, a manual joystick and first-person-view steering setup can be used to collect manual reckless driving. The ability to deploy autonomously with multiple vehicles enables large-scale collection of driving, for a total of over 3 hours of driving data. Additionally, we augment the dataset of collected trajectories by applying a sliding window to generates additional sequences for training.



(a) Nominal Driving (b) Lane Detector Failure

Figure 3: FailureNet distinguishes between nominal drivers and reckless drivers. Sequence of poses (green line) input into the network and outputs a prediction of vehicle failure (red/green sphere).

Table 1: FailureNet Accuracy on Validation Data

Method	# learnable parameters	Accuracy in % All	Periodic	Lane Shift	Manual	Speeding	Nominal
Speed Threshold	0	70.68	83.59	7.22	0.33	100.00	99.37
Speed + MLP	5,569	74.86	97.95	37.91	32.67	100.00	88.10
Kalman Filter	0	60.29	92.30	75.86	81.48	100.00	36.79
FFT Threshold	0	55.40	0.0	0.00	5.67	4.73	99.19
FFT + MLP	6,209	93.00	95.38	92.06	73.67	100.00	97.11
MLP	8,129	97.44	96.41	97.11	93.67	100.00	98.38
FailureNet-LSTM	26,049	98.42	97.44	98.19	96.33	99.32	99.10
FailureNet-GRU	21,633	97.78	94.36	95.67	95.00	98.65	99.55
FailureNet-CfC	**1,936**	97.78	92.82	100.00	92.33	98.65	99.46

Detecting Failures and Warning Cross Traffic The intelligent traffic lights in the city monitor the oncoming traffic positions and warn incoming traffic if anomalies are detected. The traffic manager accesses the pose information published from the motion capture, and FailureNet receives each vehicle's pose at 2Hz and inputs a sequence of L previous poses into the RNN. Further details on the network architecture can be found in Appendix B. If FailureNet's output is above a detection threshold $\bar{Z}$, then a warning is sent to AVs approaching the intersection.

4 Results

Model Accuracy on Validation Data In Table 1, we report the detection accuracy for each method on both the entire validation dataset (failure modes and nominal drivers), as well as accuracy in detecting each individual failure mode. The RNN architectures (LSTM, GRU, CfC) provide the highest accuracy rates over the baselines. For baseline details, see Appendix C. FailureNet-LSTM is overall the best performing, with highest accuracy on the most difficult failure mode (manual driver). FailureNet-GRU and FailureNet-CfC provide the highest true negative rate on the nominal driver validation data. One advantage of FailureNet-CfC is its relatively small size compared to the other RNNs and MLP (which require $10\times$ and $4\times$ the parameters, respectively).

Safety Evaluation in the Miniature City Finally, we deploy FailureNet in the hardware testbed with two vehicles, one that drives nominally and one that drives with one of the failure mode activated. Figure 3 shows the detector deployed in the testbed, with the input sequence and prediction visualized. We deploy each method and failure mode for 3 minutes each and evaluate the accuracy of the detector, running at 1Hz. In Table 2, we report the accuracy for various baselines and RNNs, in various failure settings. We find that FailureNet-LSTM and Failurenet-CfC perform best deployed, with an accuracy of 84%. The speed threshold performs well on the failure modes with speed components, however, fails to identify the lane shift failure mode since speed is unaffected. In contrast, our approach performs well across all failure modes and outperforms the MLP when evaluated online.

Table 2: FailureNet Accuracy Deployed in Hardware Testbed

Method	All	Periodic	Lane Shift	Manual	Speeding	Nominal
Speed Threshold	73	100	8	98	100	100
FFT Threshold	21	13	1	7	2	98
Kalman Filter	71	75	78	92	94	21
MLP	74	65	75	79	64	88
FailureNet-LSTM	**84**	79	90	95	69	84
FailureNet-GRU	79	56	88	86	64	95
FailureNet-CfC	**84**	79	87	78	85	87

References

[1] National Center for Statistics and Analysis, "Speeding: 2020 data, Tech. Rep. June, 2022.

[2] N. Buckman, A. Hansen, S. Karaman, and D. Rus, "Evaluating Autonomous Urban Perception and Planning in a 1/10th Scale MiniCity," *Sensors*, vol. 22, no. 18, p. 6793, sep 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/18/6793

[3] S. Hecker, D. Dai, and L. Van Gool, "Failure Prediction for Autonomous Driving," *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2018-June, no. Iv, pp. 1792–1799, 2018.

[4] J. Svegliato, K. H. Wray, S. J. Witwicki, J. Biswas, and S. Zilberstein, "Belief Space Metareasoning for Exception Recovery," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, nov 2019, pp. 1224–1229. [Online]. Available: https://ieeexplore.ieee.org/document/8967676/

[5] M. Zhang, C. Chen, T. Wo, T. Xie, M. Z. A. Bhuiyan, and X. Lin, "SafeDrive: Online Driving Anomaly Detection From Large-Scale Vehicle Data," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 2087–2096, 2017.

[6] D. A. Johnson and M. M. Trivedi, "Driving style recognition using a smartphone as a sensor platform," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pp. 1609–1615, 2011.

[7] I. Vasconcelos, R. O. Vasconcelos, B. Olivieri, M. Roriz, M. Endler, and M. C. Junior, "Smartphone-based outlier detection: a complex event processing approach for driving behavior detection," *Journal of Internet Services and Applications*, vol. 8, no. 1, 2017.

[8] G. C. M. Quintero, J. A. O. Lopez, and J. M. P. Rua, "Intelligent erratic driving diagnosis based on artificial neural networks," in *2010 IEEE ANDESCON*. IEEE, sep 2010, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/5631576/

[9] A. Siddiqui, A. Fern, T. G. Dietterich, and S. Das, "Finite Sample Complexity of Rare Pattern Anomaly Detection," in *UAI'16: Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, 2016, pp. 686–695.

[10] T. Wu and J. Ortiz, "RLAD: Time Series Anomaly Detection through Reinforcement Learning and Active Learning," mar 2021. [Online]. Available: http://arxiv.org/abs/2104.00543

[11] C. Ryan, F. Murphy, and M. Mullins, "End-to-End Autonomous Driving Risk Analysis: A Behavioural Anomaly Detection Approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1650–1662, 2021.

[12] H. Kawashima, F. Oba, S. Control, D. Group, and A. Shinko, "A Multi-Model Based Fault Detection and Diagnosis of," no. October, pp. 1–6, 2003.

[13] Z. H. Duan, Z. X. Cai, and J. X. Yu, "Fault diagnosis and fault tolerant control for wheeled mobile robots under unknown environments: A survey," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2005, no. April, pp. 3428–3433, 2005.

[14] Y. Takei and Y. Furukawa, "Estimate of driver's fatigue through steering motion," *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, no. 1, pp. 1765–1770, 2005.

[15] G. Di Biase, H. Blum, R. Siegwart, and C. Cadena, "Pixel-wise Anomaly Detection in Complex Driving Scenes," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 16 913–16 922, 2021.

[16] A. Doshi and M. M. Trivedi, "On the roles of eye gaze and head dynamics in predicting driver's intent to change lanes," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 3, pp. 453–462, 2009.

[17] L. Fletcher and A. Zelinsky, "Driver Inattention Detection based on Eye Gaze–Road Event Correlation," *International Journal of Robotics Research*, vol. 28, no. 6, pp. 774–801, 2009.

[18] J. Morton, T. A. Wheeler, and M. J. Kochenderfer, "Analysis of Recurrent Neural Networks for Probabilistic Modeling of Driver Behavior," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1289–1298, 2017.

[19] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-Feasible Trajectory Forecasting With Heterogeneous Data," 2020. [Online]. Available: http://arxiv.org/abs/2001.03093

[20] M. S. Shirazi and B. T. Morris, "Looking at Intersections: A Survey of Intersection Monitoring, Behavior and Safety Analysis of Recent Studies," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 1, pp. 4–24, 2017.

[21] G. M. Björklund and L. Åberg, "Driver behaviour in intersections: Formal and informal traffic rules," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 8, no. 3, pp. 239–253, 2005.

[22] J. Sun, S. Kousik, D. Fridovich-Keil, and M. Schwager, "Self-Supervised Traffic Advisors: Distributed, Multi-view Traffic Prediction for Smart Cities," *arXiv preprint*, 2022.

[23] D. J. Phillips, T. A. Wheeler, and M. J. Kochenderfer, "Generalizable intention prediction of human drivers at intersections," *IEEE Intelligent Vehicles Symposium, Proceedings*, no. Iv, pp. 1665–1670, 2017.

[24] S. Lefèvre, C. Laugier, and J. Ibañez-Guzmán, "Risk assessment at road intersections: Comparing intention and expectation," *IEEE Intelligent Vehicles Symposium, Proceedings*, pp. 165–171, 2012.

[25] F. D. Salim, S. W. Loke, A. Rakotonirainy, B. Srinivasan, and S. Krishnaswamy, "Collision pattern modeling and Real-Time collision detection at road intersections," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pp. 161–166, 2007.

[26] H. Kowshik, D. Caveney, and P. R. Kumar, "Provable systemwide safety in intelligent intersections," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 3, pp. 804–818, 2011.

[27] K. Dresner and P. Stone, "Mitigating catastrophic failure at intersections of autonomous vehicles," *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, vol. 3, pp. 1361–1364, 2008.

[28] B. Yu, S. Bao, F. Feng, and J. Sayer, "Examination and prediction of drivers' reaction when provided with V2I communication-based intersection maneuver strategies," *Transportation Research Part C: Emerging Technologies*, vol. 106, no. November 2018, pp. 17–28, 2019.

[29] Y. Feng, C. Yu, S. Xu, H. X. Liu, and H. Peng, "An Augmented Reality Environment for Connected and Automated Vehicle Testing and Evaluation," *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2018-June, no. Iv, pp. 1549–1554, 2018.

[30] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[31] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," pp. 1–9, 2014. [Online]. Available: http://arxiv.org/abs/1412.3555

[32] M. Lechner, R. Hasani, M. Zimmer, T. A. Henzinger, and R. Grosu, "Designing worm-inspired neural networks for interpretable robotic control," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 87–94.

[33] M. Lechner, R. Hasani, A. Amini, T. A. Henzinger, D. Rus, and R. Grosu, "Neural circuit policies enabling auditable autonomy," *Nature Machine Intelligence*, vol. 2, no. 10, pp. 642–652, 2020.

[34] C. Vorbach, R. Hasani, A. Amini, M. Lechner, and D. Rus, "Causal navigation by continuous-time neural networks," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[35] R. Hasani, M. Lechner, A. Amini, D. Rus, and R. Grosu, "Liquid time-constant networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 9, pp. 7657–7666, May 2021.

[36] A. Gu, K. Goel, and C. Re, "Efficiently modeling long sequences with structured state spaces," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=uYLFoz1vlAC

[37] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Advances in neural information processing systems*, 2018, pp. 6571–6583.

[38] R. Hasani, M. Lechner, A. Amini, L. Liebenwein, M. Tschaikowski, G. Teschl, and D. Rus, "Closed-form continuous-depth models," *arXiv preprint arXiv:2106.13898*, 2021.

[39] K. J. Friston, L. Harrison, and W. Penny, "Dynamic causal modelling," *Neuroimage*, vol. 19, no. 4, pp. 1273–1302, 2003.

A Related Works

Monitoring Ego Driver The ability to detect failures in AV stacks or anomalies in human drivers is crucial for trust in AVs. Recent work (3; 4) has explored methods for introspective monitoring of the AV stack for faults and anomalies by observing the state of the vehicle. For human drivers, neural networks learn from on-board vehicle diagnostics to identify driver anomalies (5) and (6; 7) use onboard cellphone data to train a network to identify different driving styles. (8) use a simulator to generate erratic driving and detect anomalies with a neural network. Other learning-based approaches use supervised learning (9) or reinforcement learning (10) to detect rare events in time-series data. In (11), a Gaussian Processes models nominal human driver based on pre-recorded human driver trajectories, and identify anomalies in an AV if observed steering is outside a 95% confidence interval. Non-learning approaches include identifying faults with a Kalman filter (12; 13) and analyzing the frequencies in driver steering (14) to identify driver fatigue in a simulation. In all these methods, the network requires access to the vehicle's internal state, from driver inputs to software outputs, to accurately identify driver anomalies which limits monitoring.

Monitoring Surrounding Traffic For autonomous perception and planning, many systems monitor surrounding vehicles to predict the driver's state or agent's future motion. In (15), a dataset of anomalies is generated, and a detector is trained on images to identify anomalous scenes. In (16), the eyesight of other drivers is used to predict lane change intent. (17) use eye-gaze observations to predict inattention for collision avoidance.

Instead of predicting the driver's state or driving behavior, trajectory predictors predict future trajectories directly. (18) use an LSTM to predict acceleration profiles and compare to classic driver models such as the intelligent driver model (IDM). They evaluate on the NGSIM Highway dataset comparing predicting vehicle position and actual position. (19) uses a graph-based LSTM to predict dynamically feasible trajectories for robots navigating around multiple agents. In both examples, robots and agents act nominally without failures present. In addition, predicting entire trajectories during rare failures may not be necessary or possible, especially without explicitly modeling whether a failure is occurring.

Infrastructure-based Systems Intelligent intersection managers can be used to both observe traffic participants and direct drivers to prevent collisions. (20; 21) discuss various approaches for monitoring intersections. In (22), multi-camera views are fused to predict incoming traffic for an intersection. (23) uses an LSTM to predict driver intention at intersections. In both, datasets that typically only experience nominal driving behavior are used, and rarely capture dangerous driving behaviors. In (24), a deep Bayesian network is used to predict driver intentions at intersections and validated with field experiments.

Once a dangerous driver is detected, an intelligent intersection manager should actively warn oncoming traffic of dangers. (25) use a simulator to validate a collision detection algorithm for cross traffic at intersections. For preventing collisions, (26) propose a hybrid scheduler-controller to provide provably safe intersections and in (27), a supervised intelligent reservation manager modifies existing reservations in the presence of catastrophic failures. In (28; 29), full-scale cars are deployed on closed courses to evaluate human driver acceptance of V2I recommendations. However, given the inherent dangers with full-scale testing of failure modes at intersections, previous work have been deployed either purely in simulation (27) or deployed with nominal driving behaviors (28). In this work, we deploy on real hardware in a miniaturized city (Fig. 1c) with multiple autonomous vehicles of various driving behaviors. This allows us to train and deploy reckless drivers using a physical platform without similar safety concerns.

B FailureNet

We propose a learning-based approach, FailureNet, which relies purely on external pose information of each vehicle for detecting vehicle failures.

Model Architecture Our goal is to learn a function approximator

$$\hat{z}_t = f(X_t; \theta) \tag{1}$$

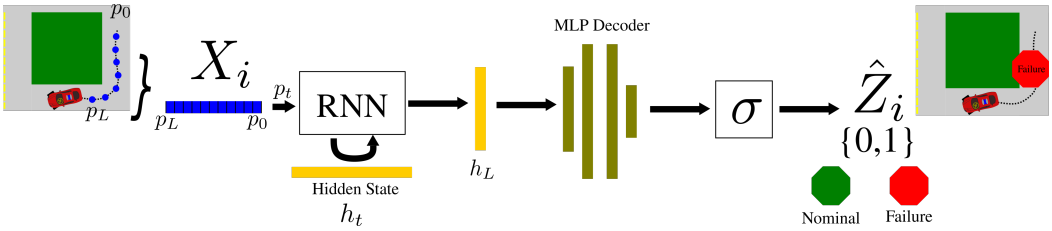


Figure 4: FailureNet Model Architecture

where $\hat{z}_t$ is the predicted state of the vehicle and X_t is the sequence of poses starting from p_t to p_{t-L}. In general, individual approximators may be deployed for each vehicle j, $\hat{z}_{j,t}$ however, for the purpose of this work, we consider estimating the status of a single vehicle and drop j for simplicity.

An end-to-end autoregressive modeling framework (Fig. 4) can be deployed to learn a proper representation from spatio-temporal input observations. To do this, we parameterize a recurrent neural network (RNN) with the following states update rule:

$$h_t = g_{RNN}(p_t, h_{t-1}) \tag{2}$$

where $h_t \in R^{n_h}$ is the hidden state of dimension n_h, and g_{RNN} is the non-linearity of the model.

We decode the hidden state through an encoder-decoder architecture, where the hidden state of the RNN compartment at the end of the input sequence, h_T, is decoded to output predictions via a multi-layer perceptron $f(\cdot)$, as follows:

$$\hat{y}_t = f(h_T) \tag{3}$$

The decoded hidden state is then passed through a sigmoidal output layer:

$$\hat{z}_t = \sigma(y_t) \tag{4}$$

where $\sigma(\cdot)$ is a logistic sigmoid function and $\hat{z}_t \in (0,1)$ corresponding to $0 = Safe$, $1 = Unsafe$.

During training, we utilize a binary cross-entropy (BCE) loss function constructed as follows:

$$\mathscr{L}(z_t, \hat{z}_t) = z_t \log(\hat{z}_t) + (1 - z_t) \log(1 - \hat{z}_t). \tag{5}$$

where z_t are the ground truth labels for whether a planning or actuation failure is occurring.

Choice of the Recurrent Neural Networks To encode input sequences, we can use gated recurrent neural networks such as the long short-term memory (LSTMs) (30) or gated recurrent units (GRUs) (31). Moreover, recent advances in end-to-end sequence modeling frameworks in robotics environments (32; 33; 34) showed the intriguing representation learning capabilities of a new class of continuous-time neural networks called liquid time-constant networks (LTCs) (35). LTCs are nonlinear state-space models (36) that are described by ordinary differential equations (ODEs) (37) or in closed-form (38) and are reduced to dynamic causal models (39), a framework through which the system can learn the cause-and-effect of a given task (34).

We use the closed-form representation of liquid neural networks, named a closed-form continuous-time neural network (CfC), as a baseline in our work to equip FailureNet with the state-of-the-art sequence modeling pipeline. CfC cells are given by the following representation (38):

$$\begin{aligned} h_t = & \sigma(-f(h_{t-1}, X_t; \theta_f)t) \odot g_1(h_{t-1}, X_t; \theta_{g_1}) + \\ & [1 - \sigma(-[f(h_{t-1}, X_t; \theta_f)]t)] \odot g_2(h_{t-1}, X_t; \theta_{g_2}). \end{aligned}$$

Here, f, g_1, and g_2 are three neural network heads with a shared backbone, parameterized by $\theta_f, \theta_{g_1}, and \theta_{g_2}$, respectively. X_t is the exogenous input to the network, t stands for input time-stamps, and $\odot$ is the Hadamard product.

C Baselines

We implement non-RNN baselines to benchmark the performance of the RNN failure estimators. The baselines include two thresholds based on filtering the input, a Kalman Filter, and a multi-layer perceptron (MLP).

Speed Threshold The vehicle speed is computed based on previous L poses and a threshold is computed based on either the average or max speed. We compute an estimate $\hat{z} = \frac{1}{L}\sum_i^L s_t \geq \bar{S}$ or $\hat{z}_{\max} = \max s_t$ where $s_t = ||x_t^2 + y_t^2||^{1/2}$. We iterate over possible thresholding values and choice of maximum or average speed, and choose a threshold that maximizes overall validation accuracy.

Fast Fourier Transform (FFT) Power Threshold For noisy inputs, we first compute the FFT of the trajectories, to distinguish between noise profiles applied at the steering and speed. We compute the one-dimensional FFT of the vehicle yaw and select the higher-order modes for thresholding, $\omega_2 \dots \omega_{L/2}$. We choose a threshold on the maximum or average spectral power ($P(\omega_k) = |\omega_k|^2$) of the sequence, by searching through possible thresholds $\bar{P}$ which produces the highest validation accuracy.

Kalman Filter A Kalman filter is also applied to noisy vehicle trajectories to evaluate the failure of the trajectory. To be specific, the failure is evaluated by setting a threshold δ_κ to the measurement post-fit residual of the Kalman filter, which is absolute distance between the observation and filter predicted position. The Kalman state is in dimension 6, which includes 3-dimensional position/orientation and corresponding velocities. The optimal threshold δ_κ is 0.2, which was found by grid search.

Multi-Layer Perceptron (MLP) We explore three different multi-layer perceptrons (MLP) with varying inputs to classify the failure state. We tune a standard MLP to determine the number of hidden layers, dimension, and dropout to maximize accuracy with the input being a concatenation of the poses in the previous L timesteps. We train two additional networks, with the same MLP architecture, but add a pre-filter at the network input which computes either the speeds s_t or FFT of the inputs ω_k.