Real-Time Anomaly Detection for IndyCar Datasets with LSTM

Yafei Wang
wangyafe@iu.edu
Department of Intelligent Systems Engineering,
Indiana University Bloomington
Bloomington, Indiana

Figure 1: Indy 500 race

1 INTRODUCTION

The IndyCar series is a very famous open-wheel car racing event in US (`https://www.indycar.com/`), where total 33 teams take part in the match. In order to better monitor and communicate between drives and team support members, hundreds of sensors and actuators are installed on the car to record and transmit many metrics, such as speed, engine rpm, gear, lap distance and brake etc, with collecting huge datasets after the events.

In order to reduce or prevent some anomaly events (such as car crash) happening in the race, the recorded datasets can be used to analyze or predict the future frame. It is difficult to obtain a good model with some pre-trained static networks in IndyCar case, as the Indycar datasets is a real-time or steaming over time. Therefore, it is important to learn the datasets with a dynamic algorithm in time series.

In this project, the state-of-the-art deep learning method-Long short-term memory (LSTM) architecture is used as DL framework to build a model, and then train the log datasets with multi-features (variables) to obtain a speed-predict model, and lastly compare the predict speed with ground truth to determine if something anomaly would happen.

2 RELATED WORK

Anomaly detection of real-time data

Anomaly detection is defined as patterns in data does not match with expected behavior. Anomaly detection is an important problem that has been studied in many research areas and applied in a lot of domains [2], in which a huge majority of cases use supervised (e.g. SVM and decision trees) and unsupervised (e.g. clustering) for batch data processing. Real-time big data processing and detect anomalies in networks has attracted much attention in recently years [4, 7]. Algorithms for real-time anomaly detection mainly include Hierarchical Temporal Memory Algorithm (HTM), Skyline, Twitter ADVec, KNN CAD, Relative Entropy, Windowed Gaussian. For example, Widanage et al. built a Hierarchical Temporal Memory Algorithm (HTM) network to annotate Indy500 dataset on anomalies with known events and evaluate the performance of detection [12]. With the rapid development of AI, Deep Learning is playing a vast role in anomaly detection [1].

Long short-term memory (LSTM) network

LSTM is an artificial recurrent neural network (RNN), which has already been used in many deep learning areas for exploring sequences of data since was introduced in 1997 [5]. LSTMs are explicitly designed to reduce the long-term dependency problem which RNNs particular have. Instead having a single neural network layer in RNNs, LSTMs have four layers and interact in a complicated way, see Figure 2.

Unlike standard feedforward neural networks, LSTM has feedback connections. The inherent properties of LSTM makes it an ideal algorithm for anomaly detection tasks involving time-series, non-linear numeric streams of data, for example, some recently works used LSTM to obtain good results for anomaly detection in spacecraft operation and video detection research [8, 9].

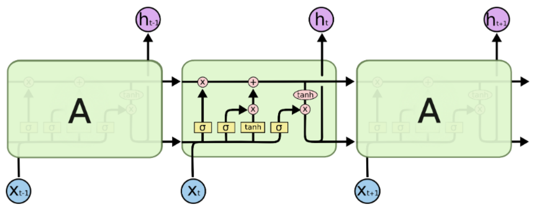


Figure 2: The schematic of LSTM network [3]

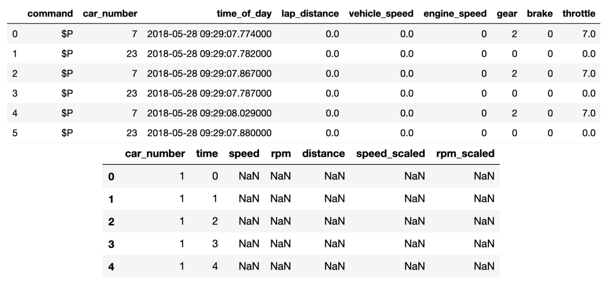
The application of LSTM in anomaly detection

Previous work have demonstrated that LSTM/RNN network is capable of addressing multivariate time-series data without the need for dimensionality reduction [10], which is very important for IndyCar case due to multivariate real-time metrics for each car such as speed, engine rpm, gear, lap distance and brake etc. Here, we surveyed some open-source projects in github which deployed LSTM to do anomaly detection in real-time data series:

`https://github.com/khundman/telemanom` [6],
`https://github.com/akash13singh/lstm_anomaly_thesis` [11],
`https://github.com/Lzeey/time-series-anomaly`,

3 DATASET PRE-PROCESSING

The dataset used in this project is *IPBroadcaster_Input_201805-27_0.log* provided by IndyCar Series. There are 33 cars involved in the race, with 24 cars finished match in 2018. The original dataset has column *command, car_number, time_of_day, lap_distance, vehicle_speed, engine_speed, gear, brake, throttle* etc different columns, see Figure 3. In the pre-processing, original raw dataset is cleaned by rearranging the row with *$P* and stacking different cars from 1st to 33rd.



	command	car_number	time_of_day	lap_distance	vehicle_speed	engine_speed	gear	brake	throttle
0	$P	7	2018-05-28 09:29:07.774000	0.0	0.0	0	2	0	7.0
1	$P	23	2018-05-28 09:29:07.782000	0.0	0.0	0	0	0	0.0
2	$P	7	2018-05-28 09:29:07.867000	0.0	0.0	0	2	0	7.0
3	$P	23	2018-05-28 09:29:07.787000	0.0	0.0	0	0	0	0.0
4	$P	7	2018-05-28 09:29:08.029000	0.0	0.0	0	2	0	7.0
5	$P	23	2018-05-28 09:29:07.880000	0.0	0.0	0	0	0	0.0

	car_number	time	speed	rpm	distance	speed_scaled	rpm_scaled
0	1	0	NaN	NaN	NaN	NaN	NaN
1	1	1	NaN	NaN	NaN	NaN	NaN
2	1	2	NaN	NaN	NaN	NaN	NaN
3	1	3	NaN	NaN	NaN	NaN	NaN
4	1	4	NaN	NaN	NaN	NaN	NaN

Figure 3: Screenshot of raw dataset (above) and cleaned (below)

Then convert the *csv* datatype to *matrix*, and fill *NaN* with *0*. In the cleaned dataset, there are 7 multivariables, but only *speed* and *lap_distance* are used in the network training. After cleaning and extracting two features, dataset is normalized in order to speed up and improve accuracy. Lastly, dataset is used to create *trainX, trainY, testX, testY* for the train and validation, see details in source code. In the project, the first 10% of dataset is used as training and following another 5% as test.

4 MODEL ARCHITECTURE

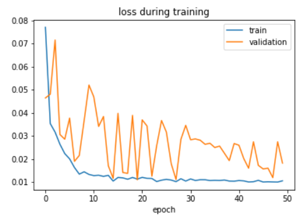
In this project, *Keras* (`https://keras.io/`) is used to build a LSTM network. The core architecture is listed as following: with two hidden layers (each with 32 units), epochs=50, batch_size=100. In the project, the train and predict results are compared and discussed with different *timestep* or *look_back*.

```
from keras.models import Sequential
from keras.layers import Dense, LSTM
model = Sequential()
model.add(LSTM(32, input_shape=(look_back, features),
return_sequences=True))
model.add(LSTM(32))
model.add(Dense(1))
model.compile(loss='mae', optimizer='adam')
model.summary()
history = model.fit(x_train, y_train, validation_data =
(x_test,y_test), epochs=50, batch_size=100, shuffle=False)
```

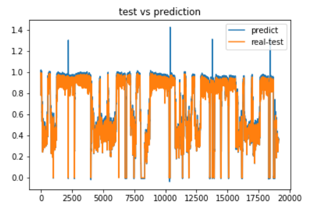
5 RESULTS

As discussed in the Section 4, the train and predict results are compared and discussed with three different *timestep* or *look_back* (10, 50, 100).

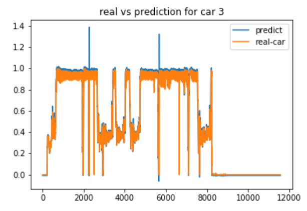
From Figure 4 (timestep=10), the train and validation loss decrease with epoch increasing, even validation is unstable. The speed ground truth of test and prediction is close. However, it is noted prediction is beyond 1.0 in certain time, which would be improved with increasing *timestep*. In addition, we selected car *3* and car *66* as examples to test the trained model (we need to point out car *66* successfully finished the match, while car *3* did not). From the results, the trained model shows good prediction results for both car *3* and car *66* in terms of speed. Here, we also set a speed difference threshold between prediction and ground truth, in which if the absolutely difference is greater than defined threshold, we determine it's anomaly (with value as 1).



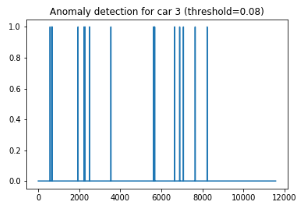
(a) training loss



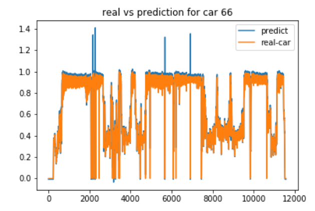
(b) test vs prediction



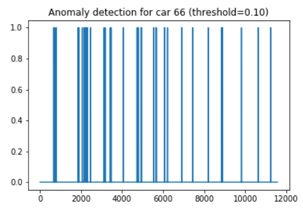
(c) ground truth vs prediction of car 3



(d) anomaly detection of car 3



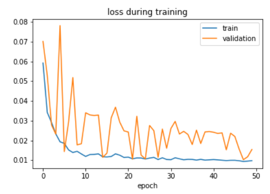
(e) ground truth vs prediction of car 66



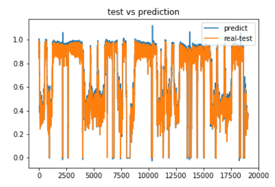
(f) anomaly detection of car 66

Figure 4: The train and predict results with timestep=10

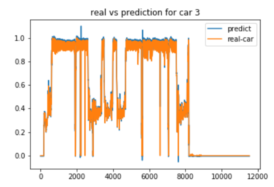
From Figure 5 (timestep=50) and Figure 6 (timestep=100), we can find that both of train loss and validation loss decrease with epoch increasing. Similar as the result of *timestep=10*, the predict speed is very close to ground truth in greater *timestep*. In addition, both predict and ground truth for car *3* and car *66* also match well in greater *timestep*. Here, it is noticed that the problem that predict speed is greater than 1.0 is alleviated with increasing the *timestep*. For the anomaly detection, we obtain the results, but no obviously difference with different *timestep* is observed here.



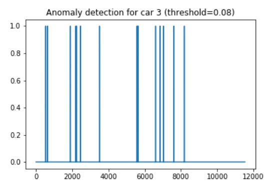
(a) training loss



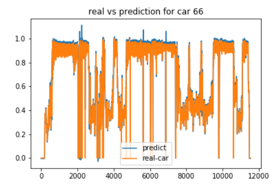
(b) test vs prediction



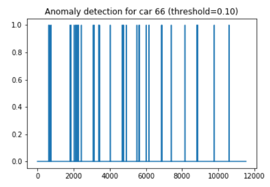
(c) ground truth vs prediction of car 3



(d) anomaly detection of car 3



(e) ground truth vs prediction of car 66



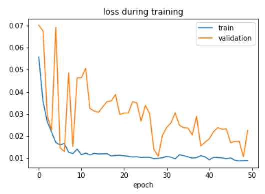
(f) anomaly detection of car 66

Figure 5: The train and predict results with timestep=50

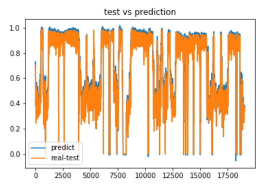
6 DISCUSSION

In this project, LSTM network is used as DL framework to build a model to train and predict with real-time, multi-features IndyCar datasets. We expect to compare the predict speed with ground truth to determine if something anomaly would happen. However, it is not a satisfied result for anomaly detection as car *3* failed was not detected with this model. Here, we conclude as we have found and propose some open questions or discussion.

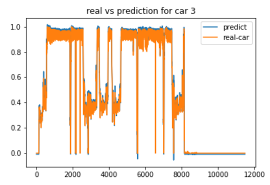
- From the results, the predict speed is very close to the ground truth
- By increasing *timestep*, the problem that prediction exceed 1.0 is alleviated
- Some open discussion
  - If car location information is available, it would be useful to obtain more accurate results
  - The image datasets is possible used to do augmentation, and then combine telemetry datasets to build model



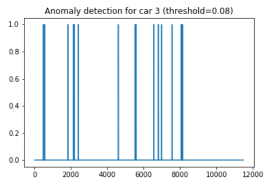
(a) training loss



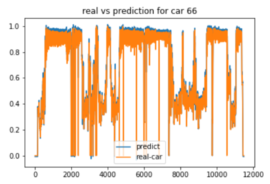
(b) test vs prediction



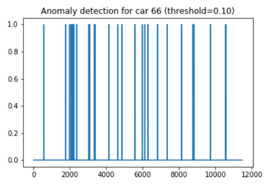
(c) ground truth vs prediction of car 3



(d) anomaly detection of car 3



(e) ground truth vs prediction of car 66



(f) anomaly detection of car 66

Figure 6: The train and predict results with timestep=100

  - Other possibility such as labelling car crashing should be more effective (see Figure 7), but it's time consuming to manually label it. What's more, only 9 cars didn't finish in 2018, so the crash datasets may be too small? Possible some techniques such as GAN could be used to generate more new data?

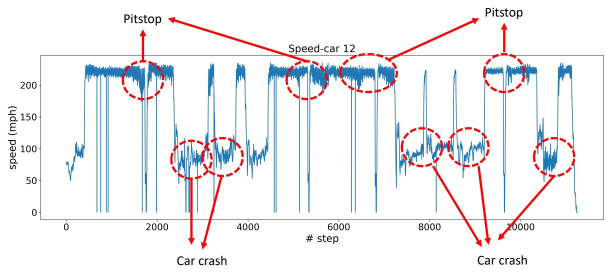


Figure 7: The labelling of anomaly events of car 12

ACKNOWLEDGEMENTS

We are very grateful for instructor Selahattin's help for cleaning the raw datasets, and thank for Prof. Judy Fox's insightful discussion. We also acknowledge access to FutureSystems at Digital Science Center, Luddy School of Informatics, Computing, and Engineering at Indiana University Bloomington.

REFERENCES

[1] Raghavendra Chalapathy and Sanjay Chawla. 2019. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407* (2019).

[2] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)* 41, 3 (2009), 1–58.

[3] Colah. 2015. *Understanding LSTM Networks*. Retrieved August 27, 2015 from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[4] Riyaz Ahamed Ariyaluran Habeeb, Fariza Nasaruddin, Abdullah Gani, Ibrahim Abaker Targio Hashem, Ejaz Ahmed, and Muhammad Imran. 2019. Real-time big data processing for anomaly detection: A Survey. *International Journal of Information Management* 45 (2019), 289–307.

[5] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[6] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. 2018. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 387–395.

[7] Mehmet Necip Kurt, Yasin Yilmaz, and Xiaodong Wang. 2020. Real-time nonparametric anomaly detection in high-dimensional settings. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[8] Weixin Luo, Wen Liu, and Shenghua Gao. 2017. Remembering history with convolutional lstm for anomaly detection. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 439–444.

[9] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. 2016. LSTM-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148* (2016).

[10] Anvardh Nanduri and Lance Sherry. 2016. Anomaly detection in aircraft data using Recurrent Neural Networks (RNN). In *2016 Integrated Communications Navigation and Surveillance (ICNS)*. Ieee, 5C2–1.

[11] Akash Singh. 2017. Anomaly detection for temporal data using long short-term memory (lstm).

[12] Chathura Widanage, Jiayu Li, Sahil Tyagi, Ravi Teja, Bo Peng, Supun Kamburugamuve, Dan Baum, Dayle Smith, Judy Qiu, and Jon Koskey. 2019. Anomaly detection over streaming data: Indy500 case study. In *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*. IEEE, 9–16.