COSE474-2022F: Final Project
“Detecting anomalies in in-vehicle network with Deep Learning”

2018320239 Hojun Ryu

1. Introduction

1.1. Motivation

In-vehicle networks, CAN(Controller Area Network), is channel for every signal in controlling vehicle activities. Controls signals like power signals, acceleration signals, front panel signals use this channel in controlling vehicle. It can be only accessed from physical channel but also can be accessed via other connected devices. Thus it is important to predict which signal should appear and detect which signal is malicious and detect which type of attack the signal is in further. With deep learning, we can make some models learn normal state CAN packets and classify which packet is malicious.

1.2. Problem definition

Problem is to learn normal state CAN packets and classify packet into normal state packet or malicious packet.

2. Methods

2.1. Backgrounds

In vehicle, each device(e.g. engine, break, front panel volume button) using CAN have its own unique ID and devices can be distinguished with it. Also CAN packets are broadcasted in CAN bus that Intrusion Detecting System(IDS) can detect CAN packet flows.
In normal state, every device send and response in specific order. Using this characteristic of CAN, we can predict which device ID should appear in context of CAN device ID.
However, CAN ID are dependent to vehicle type. It can differ from vehicle to vehicle that models should be learned with packets from only one vehicle.

2.2. Model

Recurrent Neural Network(RNN) has benefits in learning context in terms of time or sequence because each node has recurrent connections. This characteristic is important in this method because CAN packets are time series data which is appropriate to be learnt with RNN.
However, traditional RNN network can’t learn in long term. So, in this method, Long Short-Term Memory(LSTM) Autoencoder will be used in learning model. Model is like Figure 1.

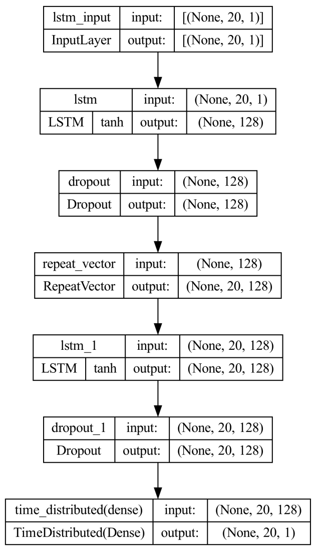

lstm_input | input: | [(None, 20, 1)]
InputLayer | output: | [(None, 20, 1)]
lstm | input: | (None, 20, 1)
LSTM | tanh | output: | (None, 128)
dropout | input: | (None, 128)
Dropout | output: | (None, 128)
repeat_vector | input: | (None, 128)
RepeatVector | output: | (None, 20, 128)
lstm_1 | input: | (None, 20, 128)
LSTM | tanh | output: | (None, 20, 128)
dropout_1 | input: | (None, 20, 128)
Dropout | output: | (None, 20, 128)
time_distributed(dense) | input: | (None, 20, 128)
TimeDistributed(Dense) | output: | (None, 20, 1)

Figure 1. Model used in this project

2.3. Algorithm

First, given dataset should be splited into specific time step because whole dataset is too long for train and testing. With some fixed time step, trainset and testset will be generated from normal dataset and DoS dataset.
With given normal trainset, train model with LSTM autoencoder by encoding and decoding. In this training process, reconstruction error should be calculated in Mean Absolute Error(MAE). With this given reconstruction errors, threshold for classifying anomaly will be max value of them.
Finally, test malicious dataset with trained model and calculate MAE of prediction error. Specific algorithm is in Algorithm 1.

Algorithm 1 Anomaly detection with LSTM autoencoder

```
Require: $time\_step > 0$
 1: // Split Dataset into time step
 2: for i ← 0 to (# of dataset − time_step) do
 3:     $trainset \leftarrow dataset[i : i + time\_step]$
 4:     $testset \leftarrow dataset[i : i + time\_step]$
 5: end for
 6: // Train model
 7: for $epoch = 1\, to\, 100$ do
 8:     for $x \in trainset$ do
 9:         $representation \leftarrow Encode(x)$
10:         $y \leftarrow Decode(representation)$
11:         Train model $s.t.$ $y$ fits to $x$
12:     end for
13:     if No change in loss for 3 epochs then
14:         Stop Training
15:     end if
16: end for
    // Calculate Threshold
17: for $x \in trainset$ do
18:     $representation \leftarrow Encode(x)$
19:     $y \leftarrow Decode(representation)$
20:     $AbsErr \leftarrow (y - x)$
21: end for
22: $threshold = max(AbsErr)$
23: // Detect Anomaly
24: for $x \in testset$ do
25:     $representation \leftarrow Encode(x)$
26:     $y \leftarrow Decode(representation)$
27:     if $(y - x) > threshold$ then
28:         classify as anomaly packet
29:     else
30:         classify as normal packet
31:     end if
32: end for
```

2.4. Score

To compare performance with other trials, accuracy was used as anomaly detection score and equation is like below.

$$accruacy = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

TP,TN,FP,FN each stands for True Positive, True Negative, False Positive, False Negative.

3. Experiments

3.1. Dataset

Table 1 is part of DoS dataset. Dataset can be accessed online and CAN packets are collected from one vehicle(KIA soul)(Seo et al., 2018). Dataset consist of Normal state, gear attack, RPM attack, DoS attack and each attack dataset consists of normal state packets and malicious packets. Normal packets are labeled in 'R' and malicious packets are labeled in 'T' that we can validate accuracy of model with this label. In this project, only Normal state and DoS attack packets are used in training and testing each. Also, learning environment resource was limited that only first 200,000 DoS packets were used in testing.
Columns consist of Timestamp, ID, DLC, Data, and label in only attack dataset. Only Timestamp and ID, and label column will be used in this method.

Type	Normal state	DoS attack
# of Packets	988,987	3,665,771
# of Normal Packets	988,872	3,078,250
# of Malicious Packets	0	587,521

Table 1. Dataset of CAN packets

Timestamp	ID	Label
1478198415.443339	0000	T
1478198415.443588	0153	R
1478198415.443832	0000	T
1478198415.444090	0002	R
1478198415.444333	0000	T
1478198415.444583	04f0	R

Table 2. Malicious dataset example

3.2. Environment

Experimented on Google Colab pro with premium GPU.

3.3. Initial trial

In initial trial, time step was set to 20 because ID peak is repeated within 20 packets as in Figure 2.

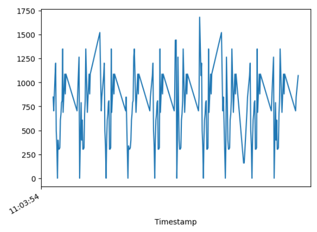


Figure 2. Normal dataset plot of 200 packets.

With trained model, MAE loss with predicting from Normal dataset and DoS dataset was like Figure 3. Max MAE loss in (a) will be threshold for detecting anomalies and histogram bars in (a) will be left histogram bars in (b). Therefore right histogram bars in (b) will be anomalies in our prediction.

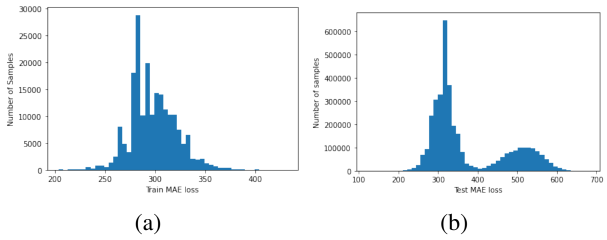


Figure 3. (a) Train MAE loss (b) Test MAE loss

Threshold and Accuracy for initial trial was 431.193 and 0.863277 each. For higher accuracy, trial with few more time step was done and Results are followed.

3.4. More trials with different time step

I tried time step of 10, 50, 100. Results are in below Figure 4 and Figure 5 and Table 3. Comparing 4 figures, train loss and test loss converges well as time step increases and accuracy is high around time step between 20 and 50.

Time step	10	20	50	100
Threshold	558.9	431.1	358.0	329.9
Accuracy(%)	83.93	86.32	86.00	84.64

Table 3. Threshold and Accuracy for different trials.

3.5. Results and SOTA

The most high accuracy of detecting malicious packet was 86.32% with 20 time step. More time step should be tested, but there was limited resource.
Also, Malicious dataset used in this project is first 200,000 packets of DoS packet. Accuracy could be different if whole malicious dataset with 3,665,771 packets.

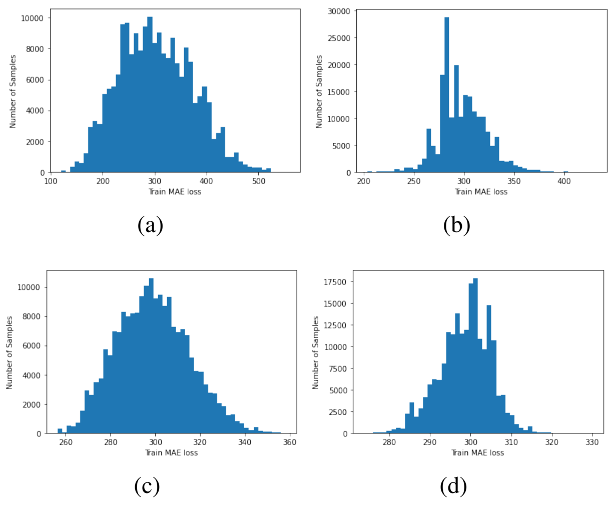


Figure 4. Train loss with time step (a) 10 (b) 20 (c) 50 (d) 100

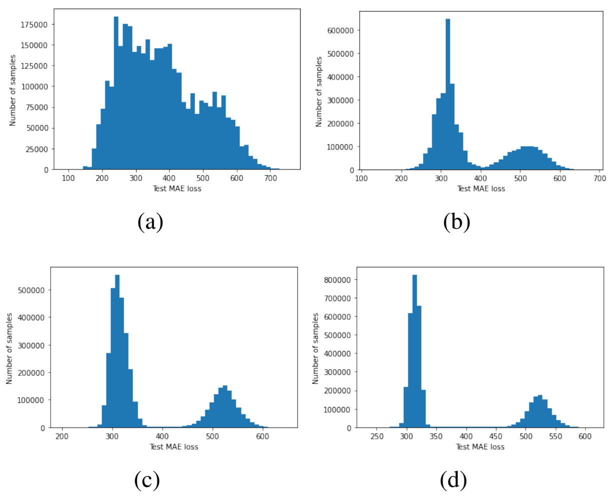


Figure 5. Test loss with time step (a) 10 (b) 20 (c) 50 (d) 100

Early stopping used in this model had 3 patience, which means training stop if loss does not change for 3 epochs. If early stopping was not used, the threshold could be calculated more precisely and accuracy could be higher.
Baseline in this project is OTIDS(Lee et al., 2017) which was first guideline of this project. And goal in this project was Rec-CNN(Desta et al., 2022). This model's accuracy was higher than OTIDS but is still less than Rec-CNN.

Time step	Accuracy
OTIDS(BaseLine)(Lee et al., 2017)	70.81
This Project	86.32
Rec-CNN(Desta et al., 2022)	98

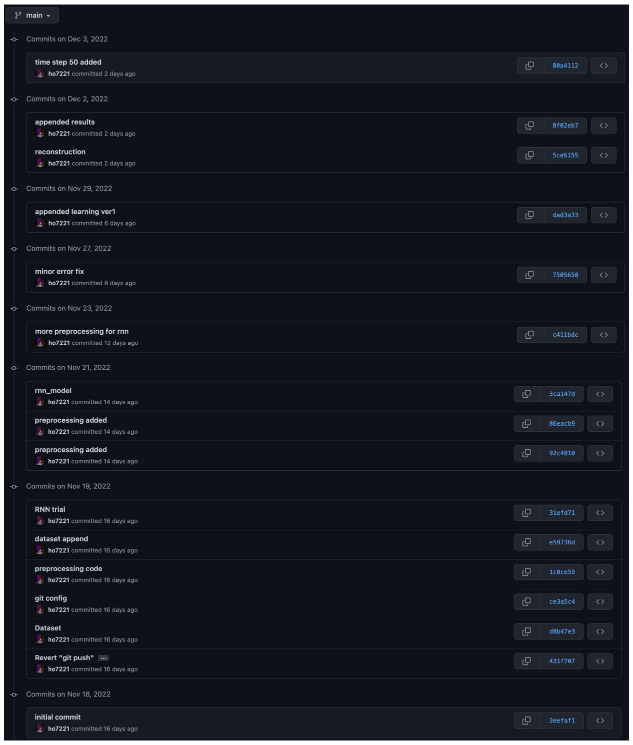
Table 4. Comparison between 3 models.

3.6. Future Experiments and directions

The most regret in this project was limited computing resource. There was no useful machine in hand, and Google Colab was too expensive that only 500 compute units were available. This is not an excuse but a pure regret. Later, with better machine in hand, more time step and more complex model can be tried for higher accuracy. Also, more dataset, not only whole DoS dataset but also Gear and RPM attack dataset, can be used to train and test accuracy in detecting anomalies for more practical performance of this project.

4. Appendix

Github address: Project Repo
Commit Log: Figure 6



```
main

Commits on Dec 3, 2022
  time step 50 added
  ho7221 committed 2 days ago                80a4112

Commits on Dec 2, 2022
  appended results
  ho7221 committed 2 days ago                8f02eb7
  reconstruction
  ho7221 committed 2 days ago                5ce6155

Commits on Nov 29, 2022
  appended learning ver1
  ho7221 committed 6 days ago                dad3a33

Commits on Nov 27, 2022
  minor error fix
  ho7221 committed 8 days ago                7505650

Commits on Nov 23, 2022
  more preprocessing for rnn
  ho7221 committed 12 days ago               c411bdc

Commits on Nov 21, 2022
  rnn_model
  ho7221 committed 14 days ago               3ca147d
  preprocessing added
  ho7221 committed 14 days ago               86eacb9
  preprocessing added
  ho7221 committed 14 days ago               92c4810

Commits on Nov 19, 2022
  RNN trial
  ho7221 committed 16 days ago               31efd71
  dataset append
  ho7221 committed 16 days ago               e59736d
  preprocessing code
  ho7221 committed 16 days ago               1c0ce59
  git config
  ho7221 committed 16 days ago               ce3a5c4
  Dataset
  ho7221 committed 16 days ago               d8b47e3
  Revert "git push" ...
  ho7221 committed 16 days ago               431f707

Commits on Nov 18, 2022
  initial commit
  ho7221 committed 16 days ago               3eefaf1
```

Figure 6. Commit Log

References

Desta, A. K., Ohira, S., Arai, I., and Fujikawa, K. Rec-cnn: In-vehicle networks intrusion detection using convolutional neural networks trained on recurrence plots. *Vehicular Communications*, 35, 2022. ISSN 2214-2096.

Lee, H., Jeong, S. H., and Kim, H. K. Otids: A novel intrusion detection system for in-vehicle network by using remote frame. *2017 15th Annual Conference on Privacy, Security and Trust (PST), Privacy, Security and Trust (PST), 2017 15th Annual Conference on, PST*, pp. 57 – 5709, 2017. ISSN 978-1-5386-2487-6.

Seo, E., Song, H. M., and Kim, H. K. Gids: Gan based intrusion detection system for in-vehicle network. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pp. 1–6, Aug 2018. doi: 10.1109/PST.2018.8514157.