Google Colab

Projet ouverture

Authors
SAGDULLIN Damir
LAIOLO Léo
TABBAH Nicola

Introduction

This project aim to recognize attacks on connected vehicles using Deep Learning (DL).

Dataset

Our dataset is quite big. Generating it takes a lot of time. This is why we saved it in a CSV file.

To discribe the treatments applied to the original dataset, we extract only a piece of the original dataset and pre-treat it. In this section, you can follow data transformation on a little portion of the dataset. At the end, we download the full pretreated dataset to train our models.

Our dataset is a part of the Vehicular Reference Misbehavior (VeReMi), built specifically for testing V2X security. We uses the received message historics from 2260 connected vehicles. Some of them send malicious messages of 19 different types.

Connected vehicles exchange lots of messages, to localize themselves or get traffic state, with other vehicles, roadside elements and infrastructures. In this study, we only work on exchanges between vehicles. This kind of message is declared with a `type:3`. So we first select all the `type:3` messages from our vehicles.

Import data

Our dataset is made of two parts:

1. The labels
2. The messages per vehicles

Labels are a list of message sender vehicle labelised as `0` for benign and `X` for malicious, where `X` corresponds to the index of the attack. We import this list of vehicles from a CSV file.

The second part of the dataset is composed of 2260 vehicles' message historics. We import them and keep only the `type:3` messages, exchanged by vehicles. Here is an example of a message content:


```
{
  "type":3,
  "rcvTime":50427.66028717679,
  "sendTime":50427.66028717679,
  "sender":57,
  "senderPseudo":10575,
  "messageID":67985,
  "pos":[983.2263964536893,910.8786100947851,0.0],
  "pos_noise":[3.6170773833523657,3.9726270832479986,0.0],
  "spd":[-8.160545717352005,-5.838995406773277,0.0],
```

```
  "spd_noise":[-0.013202827672876944,-0.00944703121125735,0.0],
  "acl":[-0.1056377948217638,-0.07553041612577578,0.0],
  "acl_noise":[0.001235094686840946,0.0021069253674089584,0.0],
  "hed":[-0.8479991671178503,-0.5299975590202584,0.0],
  "hed_noise":[24.07432760428006,19.08668548660413,0.0]
}
```


Merge BSM messages and senders labels data

Once our data imported, we centralize all the data to easily pretreat them. To do so, we merge the 2 parts of our dataset.

Because we are working on a sample of the dataset, some information might be missing. To ensure that the rest of the treatments will run properly, we delete those incomplete rows.

Split list features

We see in the tables displayed in the above section that some message data are lists of coordinates, like the position `pos`. Our model needs flatten data, not nested lists. Therefore, we split those coordinates into columns `x`, `y` and `z`, spliting `pos` into `pos_x`, `pos_y` and `pos_z`.

Drop irrelevant cols

Looking at the message data, we see that some information are irrelevant. Indeed, we are about to split our dataset on receiver vehicles. They can be identified by each one of the following data:

- `sender`, the message sender is the vehicle;
- `file`, each historic file belongs to one and only vehicle;
- `omnet_module_id`, the vehicles' id on the network didn't change during the message recording.

Once our dataset split, those columns would be filled with the same value, making it irrelevant for the model.

There are also information that couldn't be interpreted by the model and wouldn't help it to understand the messages. Those are listed below:

- `senderPseudo`, the pseudo of the vehicles are integers aiming for sender differenciation;
- `messageID`, id serve to distinguish messages that could contain the same information.

Those information are generated by the sender's communication protocols. They can be faked but are to hard to recognize, even for human. Keeping those data would brings more difficulties to understand the message than dropping it.

Finaly, the `attacker_type` indicates if the receiver vehicle is malicious (`A13`) or not (`A0`). It is a data related to the receiver vehicle so it is useless to understand received message in our case.

Scale data

To improve our model efficiency, we scale the data, excepted the labels.

Load full dataset

In the previous section, we saw all the treatments applied to a sample of the original dataset. Now, we load the complete pretreated dataset. This last one will be used to train the proposed model.

DNN

In this project, we choose to implement a classic Deep Neural Network (DNN). Indeed, we observed in the TP4 that this model was more efficient than Recurrent Neural Network. However, we didn't use Federated learning to compare this method with centralized training.

Analysis

In this section, we discuss of the performance of our model and compare it to Federated Learning (FL).

The proposed DNN achieved an accuracy score of 40 %, on the testing set. Its confusion matrice also looks partly diagonal, meaning that the model didn't confused a lot of labels, except `0` . This label stands for the benign messages. It shows that attacks are quiet difficult to recognize, even for an AI trained on more than 250 000 messages.

The proposed DNN didn't achieve 50 % of accurency on the testing dataset, though the dataset was balanced and large enough. The principal reason of this low efficiency might be the number of epochs. Indeed, we only trained our model on 50 epochs because each one of them took a lot of time. We also could have made a more complex DNN but we couldn't compare it to those we used with FL.

Federated Learning reached better results in the last project, though training were longer. Indeed, FL needed 20 epochs per DNN and a local models fusion to make one learning step with the global model. Moreover, each DNN used in FL had much less data and less classes to predict. Because the task was easier, the performance was better. To really compare the centralized learning and FL, we should have us the same dataset.

Conclusion

In this project, we implemented a DNN to recognize and classify attacks in a connected vehicular network. Our proposed model reached 40 % of accuracy with only 50 epochs, though it could have been trained longer.

To improve our project, we can compare our model with one trained on Federated Learning, using the same dataset.