CS581 Project outline – Exploring TCNs for Network Anomaly Detection.

By Sanket Mehrotra and Kaustubh Jawanjal

Application: Anomaly Detection on the Edge with RNNs, TCNs and Optimized TCNs

Models used: RNNs (LSTMs/LSTM Autoencoders), TCNs, Quantized and Weight Pruned TCNs, (maybe HTMs based on ref 4.)

Implementation:

TCNS are being hailed across the board as the successors to RNNs. We seek to try to show this ourselves in an embedded context for anomaly detection in small LANs used as in-vehicle communication networks also known as CANs. TCNs are not part of a standard keras/tensorflow API, so we will be working with a reference implementation based on the code provided by the authors of [ref 6]. LSTMs can be used via keras models, PyOD package's implementation or H2O models(PyOD - Python Outlier Detection Toolkit, H20 - Java based toolkit for data science). I propose to implement both with reference to anomaly detection in networks, a task that does require some short-term/long-term memory and compare their performance. I also plan to compare optimized TCNs as a third model in this project.

Optimization:

We will experiment with compression and optimization techniques covered in our assignment 3 - the Temporal Convolutional Networks in the reference code (ref 6 and 7) are implemented Conv1D layers, therefore almost all the techniques implemented in our assignments will be applicable to optimize these networks as well.

It will be interesting to apply these optimizations on RNNs and LSTMs. It is something we have not covered in class so I will have to understand the computations performed in these types of networks before implementing the optimizations and seeing how they affect the models.

Some would be:

- Quantization to float16

- Weight sharing

- Weight Pruning

Datasets:

1. Kitsune - Network Intrusion Detection - Mirai Botnet Dataset:

A preprocessed (feature extracted) network capture from an IoT network, where the Mirai malware (botnet) begins infecting other devices and tries to call home to its Command & Control Server after approximately 1 million packets.

Size ~ 1.27 GB - 764,000 packets.

116 Columns(extracted statistical features for each packet extracted using the AfterImage feature extractor.)

https://www.kaggle.com/ymirsky/network-attack-dataset-kitsune?

There are also other parts of this dataset representing other regular attacks such as injection, denial of service and man in the middle than can be used.

2. CAN Intrusion Dataset - https://ocslab.hksecurity.net/Dataset/CAN-intrusion-dataset

A dataset prepared by the authors of ref #3, in which they test a non-DNN based approach to detect network intrusion detection in CAN networks. I've emailed them for access to their dataset, they are yet to reply.

Size: 4 files totaling up to ~ 400 MB

On their site, they describe the dataset as follows:

This dataset is a collection of 4.4 million CAN attack related messages divided according to a specific type of attack that they simulate. The attacks covered are:

1. CAN Denial of Service attacks: Attacks in which a 0x000 can ID message is injected into the network in short cycles.
2. Fuzzy attack: Injecting messages of spoofed random CAN ID and data values.
3. Impersonation attack: Injecting messages impersonating nodes

These attack messages are mixed with regular state CAN messages that are normally exchanged in a network.

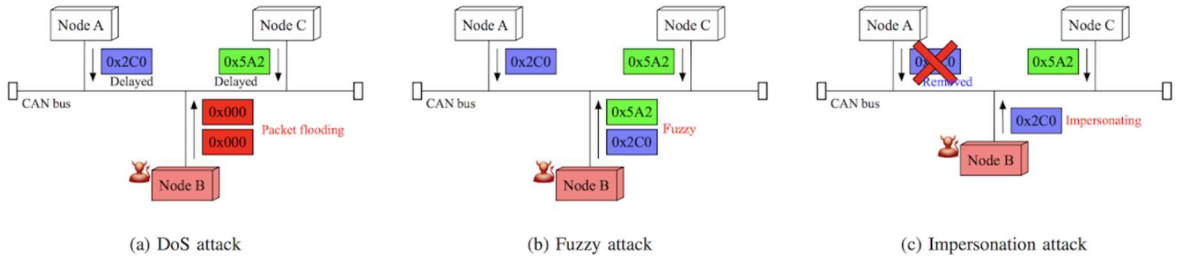

Figure 1: Types of CAN attack data Image Ref: https://ocslab.hksecurity.net/Dataset/CAN-intrusion-dataset

Attack type	# of messages
DoS Attack	656,579
Fuzzy Attack	591,990
Impersonation Attack	995,472
Attack free state	2,369,868

Things to focus on:

- Using this new type of network for the first time (not yet implemented in the standard tf/keras API).
- Short and simple implementations of the networks.
- Hyper parameter tweaking and comparison experiments.
- Large dataset means that the train-test split must be carefully calculated. It may train the network to perform better but will take a long time and may be prone to overfitting issues.
- Effect of optimizations to get a small yet well performing tflite model.
- Well documented code.

Qualitative Metrics:

True Positive Rate, False Positive Rate, Precision, Recall, F1 score, RoC curves

Ref Papers and Related work:

1. https://www.mdpi.com/2076-3417/9/15/3174/htm - DNNs in CAN network detection
2. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling (https://arxiv.org/pdf/1803.01271.pdf) - TCNs being compared to RNNs for simple tasks
3. Hyunsung Lee, Seong Hoon Jeong and Huy Kang Kim, "OTIDS: A Novel Intrusion Detection System for In-vehicle Network by using Remote Frame", PST (Privacy, Security and Trust) 2017 (https://www.ucalgary.ca/pst2017/files/pst2017/paper-67.pdf) - CAN Dataset
4. https://www.researchgate.net/publication/322814361_A_Distributed_Anomaly_Detection_System_for_In-Vehicle_Network_using_HTM - New NN technique for similar application of in-vehicle network detection.
5. http://urban-sustain.org/papers/GhoshIEEEBigData.pdf - An application of Autoencoders LSTMs for anomalous records and sequence detection.
6. https://github.com/locuslab/TCN,
7. https://github.com/philipperemy/keras-tcn#keras-tcn
8. A framework for end-to-end deep learning-based anomaly detection in transportation networks. NeemaDavis,GauravRaina,KrishnaJagannathan https://www.sciencedirect.com/science/article/pii/S2590198220300233
9. Palisade: A framework for anomaly detection in embedded systems. Sean Kauffman, Murray Dunne ,Giovani Graciolib, Waleed Khan, Nirmal Benann, Sebastian Fischmeister https://www.sciencedirect.com/science/article/pii/S1383762120301545