This past week I have been working on finding a suitable DCNN model for network threat detection. I found a model called Deep Packet Inspection (DPI) that is tailored to analyzing live network traffic. I am working on an implementation of nDPI, an open-source DPI toolkit.

nDPI Notes

• Can work with live network data from our network tap

• Unsupervised machine learning models using nDPI can identify new, unknown threat patterns in network traffic.

• Reinforcement learning with nDPI can optimize threat detection and response strategies in distributed networks.

I am doing research and working on implementing an RNN/LSTM.

Working on:

- Imports
  - Ignore warnings
  - Import dataset
  - Preprocessing
  - Trainer function
    - Define hyperparameters
      - Define/train model
      - Predict/test model
      - Generate/print metrics
  - Visualization?

The hidden state at time is a function of the input at time and the hidden state at time ī 1 ī 1.

2.

The hidden state update equation can be represented as: =(ī 1+ +) = (ī 1 + +), where is the hidden state at time , is the input at time , and are weight matrices, is the bias term, and

Conduct regular security audits and assessments of the RNN model and its infrastructure. Identify and address vulnerabilities promptly to maintain a robust security posture.

Long Short-Term Memory (LSTM) networks, a specific type of recurrent neural network (RNN) designed to overcome the vanishing gradient problem and capture long-term dependencies in sequential data:

1. LSTM networks have a more complex structure compared to traditional RNNs. They include memory cells, input gates, forget gates, and output gates, allowing them to selectively retain and update information over time.

2. The key component of LSTM is the memory cell, which can store information for long durations. The cell state ( ) carries information from previous time steps.

3. The input gate ( ) regulates how much new information should be stored in the memory

10. Multiple LSTM layers can be stacked on top of each other to form a deep LSTM network, allowing for hierarchical representation of sequential data.

11. Various LSTM variants have been proposed, such as peephole connections and coupled forget and input gates, to enhance the capabilities of the original LSTM architecture.

Collaborative Machine Learning (CML) Research:

- CML is a type of machine learning that allows multiple entities to collaborate on training and improving machine learning models.

- Also referred to as federated machine learning.

- Federated machine learning is CML without centralized training data.

- The big difference is that central machine learning ñmoves data to the computationò and federated machine learning ñmoves the computation to the data.ò

- Currently using a research blog from Google Research that focuses on Federated Learning.

- A possible framework that we can use to implement CML is Flower, listed as ña friendly federated learning framework.ò

- Flower offers a great tutorial with visuals that help aid the understanding of both machine learning and federated machine learning.

- Federated machine learning allows the use of machine learning where it wasnôt possible before.

- Other possible CML frameworks are NVFlare, FATE, and TensorFlow. I am still looking into these frameworks more in-depth.

This past week I have been working on finding a suitable DCNN model for network threat detection. I found a model called Deep Packet Inspection (DPI) that is tailored to analyz ⌞ þ MthaÒ ɐ ned to analyz ned

XGBoost (Extreme Gradient Boosting) is a machine learning algorithm that excels in classification and can easily and quickly train both binary and multiclass classification algorithms. It differs from other machine learning algorithms through its use of Decision Trees, Gradient Boosting, and Boosting rounds instead of standard machine learning processes such as epochs. The tree splits the data through binary decisions into more manageable pieces for the algorithm and the gradient boosting constructs new trees to get better results from the last session. The number of boosting rounds is the number of times this process is completed.

In our algorithm, we first take the large data set weôve been working with and slim it down to just the information that we will be getting from the Zeek-Kafka live stream so that we can train our model with the same information that we will be getting in practice. We then split the data into x and y, where x is the raw data, and y is the test results on whether or not the data was an attack or not, and what type of attack it was. The attack type data is then fed into an ordinal encoder which converts the different categories of attack into numerical representations that we can train the model on. We then do something similar with the non-numeric x data by classifying it as the type ñcategoryò. The next thing we do is split the data into train and test groups, with 70 percent of the data being used to train the model and then 30 percent of the data being used to test the model after training. The data is then put into a classification matrix and trained. The parameters we use when training are a multi:softprob as the objective, gpu_hist as the tree method, 15 different classifications, and 25 boosting rounds. After that, we can predict the test cases and measure the accuracy and precision of the model.