MITRE TC4TL Challenge System Description

Jim Houchens, Jeff Gold, Nicholas Maynard, Mark Krangle, Shreeja Kikkisetti
The MITRE Corporation
7515 Colshire Dr
McLean, VA, 22102, USA
{houchens, jgold, nmaynard, mkrangle, kikkisetti}@mitre.org

***Abstract*—This document has been created per Section 6.3.3 of the NIST Pilot TC4TL Challenge Evaluation Plan [1].**

Keywords—machine learning, artificial intelligence, COVID-19, Bluetooth, TensorFlow, NIST

I. Introduction

This document has been created under the direction of the NIST Pilot TC4TL Challenge. The intent of this document is to capture the description of the system built, the model training process used, the tuning and inference activities executed, and additional insight leveraged by the MITRE competition team so as to enable other researchers the ability to reasonably reproduce this work.

II. DATA RESOURCES

A. Data Used

The data used for training initially included the combination of the NIST provided challenge training data set and the MITRE Range Angle Structured [2] data sets. After the directive went out to no longer use the NIST provided development data (July 29, 2020), the MITRE Range Angle Structured was used exclusively. This was the only data set used until additional training data was made available via the NIST Challenge portal on August 8, 2020 and then additionally some new data created internally by the competition team as well (gathered in a similar fashion as the MITRE Range Angle Structured Dataset).

B. Data Preparation

Once the development data was identified, it had to be preprocessed in order to be used for used for model training. The primary concern is that not all sensors report their values at the same frequency, thereby creating non-uniform records over time (see Figure 1).

Figure 1 - Sensor Report Frequency

To remedy this, the MITRE team "flattened the pings" by creating a super-record that contains all values from each sensor at each time hack (the total time hack count is the count of the most frequently reporting sensor, Bluetooth Received Signal Strength Indicator (RSSI) in our case). The sensors that report less frequently continue to show previously reported values in each super-record until a new value is read in, where it is then put in place of the previous value (until a new value for that sensor is reported and replaces it). This approach allowed the MITRE team to have a consistent record with all columns present and with valid values, thereby enabling each record to be used in model creation and testing.

III. ANALYSIS AND RESULTS

The MITRE team leveraged two approaches for the NIST Challenge competition. The two approaches selected for estimating distance were random forests and neural networks.

A. Random Forest

The MITRE team found research that provided an approach to calculate distance estimation for Bluetooth RSSI leveraging random forests [3]. The initial work was done with the sklearn random forest base model, which was then improved upon utilizing Randomized Cross Validation (CV) for hyperparameter optimization. The Randomized CV nearly halved the mean average error (mae), but it was still not enough to predict values better than other approaches. It the future, estimations could possibly be improved with additional training data or further hyperparameter optimization.

B. Neural Network

In parallel to the random forest work, the MITRE team also worked with neural networks (TensorFlow with Keras) to see if neural nets could be useful to estimate distance based on Bluetooth RSSI and the additional sensor values provided in the competition. The neural network approach fared much better than random forest and that is where the MITRE team spent the majority of the time. Figure 2 below is a scatter plot from one of the earlier neural net runs comparing real values to predicted values.

Approved for Public Release; Distribution Unlimited. Public Release Case Number 20-2339

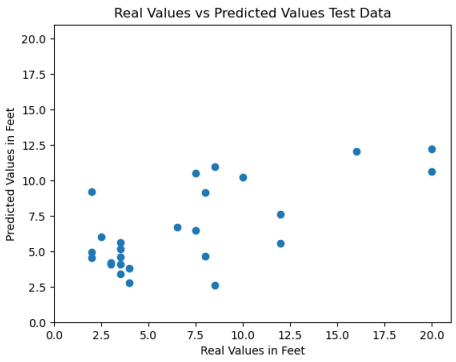

Figure 2 - Neural Net Real vs. Predicted Values

The neural net took the input data and produced an output, containing a prediction for value of the test data set. This process is similar to what is shown in Figure 3 below.

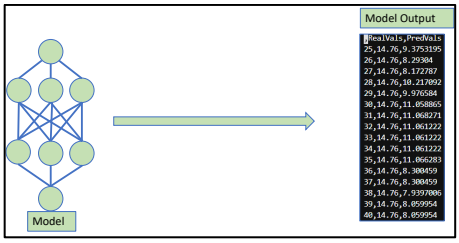

Figure 3 - Sensor Data to Distance Prediction

Once the initial neural net produced an output, the detector aggregated the estimates to predict a single distance value per collect/file. This was done in a manner shown in Figure 4 below.

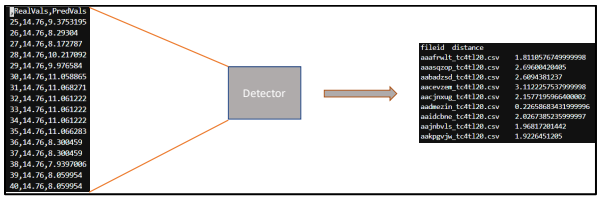

Figure 4 - Detector Summary

For our detector, we attempted several aggregations to best estimate distance per file/session. We used mean, median and a weighted average. We started with mean as it was simple to execute, but it is prone to issues when there are outliers. Then we switched to median, which had similar benefits, but is less prone to outliers. Finally, we did some analysis leveraging weighted averages to hopefully allow us to more accurately estimate distances within 6 feet (weighting those higher) as that is where the worst performance of our models tended to be.

Once the processing flow was ironed out, MLflow [4] was leveraged to automate the experiments. This enabled us to go from approximately 10 runs per day manually to over 100 runs per day, with a repository of artifacts updated from each run. The team automated runs with MLflow and varied several parameters per run, including:

Model Format

- Input Layers: Varied inputs from just the Bluetooth RSSI to the Bluetooth RSSI, Magnetometer, Accelerometer.
- Hidden Layers: varied from 1 to 10
- Output Layer: Linear
- Loss Function: Mean Squared Error or Mean Average Error

Training Parameters

- Batch Size: Varied from 1 to 2000
- Learning Rate: Varied from 0.001 to 0.01
- Epochs: Varied from 5 to 150
- Train/Test Split: Varied from 5/95 to 99/1

MLflow was used with shell scripts to iterate through the combination of the above parameter ranges. Once the pipeline was complete with the MLflow automation, we began to see an increase in accuracy in just two days of running in a fairly automated manner as shown in figures 5 and 6 below.

Best estimates done via manual execution.

Scoring Run Results #1

SUBSET	D	P_MISS	P_FA	NDCF
fine_grain	1.20	0.94	0.01	0.94
fine_grain	1.80	0.61	0.25	0.86
fine_grain	3.00	0.45	0.52	0.97
coarse_grain	1.80	0.57	0.31	0.88

Figure 5 - Best Estimates Before MLflow Automation

After two days of automated running with varied parameter configurations.

SUBSET	D	P_MISS	P_FA	NDCF
fine_grain	1.20	0.94	0.01	0.94
fine_grain	1.80	0.42	0.26	0.68
fine_grain	3.00	0.02	0.88	0.90
coarse_grain	1.80	0.35	0.25	0.60

Figure 6 - Best Estimates After MLflow Automation

Overall, training time varied from a few minutes to a few hours. However, the best predictions were created with low number of epochs, resulting in runtimes from approximately 5 minutes to 15 minutes.

IV. HARDWARE

The MITRE team executed the work for this challenge on a virtual machine within the MITRE network, using on the CPU option for TensorFlow (we did not leverage a GPU for this competition). The details of the node are as follows:

- CPU: 16 cores
- Memory: 128MB
- Disk: 140GB

- OS: CentOS 8

V. LESSONS LEARNED & NEXT STEPS

This challenge provided an excellent opportunity to mature our machine learning capabilities and better understand what was possible with Bluetooth and other sensor data for distance estimation. There are several lessons learned and potential next steps to explore for the team as we continue our work.

A. Lessons Learned

There are several things that were learned along the way that could be leveraged in a similar future activity, including the importance of validation data. After we started including the NIST data for validation of the models the results improved substantially. The team was also able to accurately see problem with our models (overfit to data). The other significant lesson learned was about the general test schedule. It was a bit challenging as the exact test schedule wasn't defined until the last week or so. This made things challenging because the automated workflow wasn't complete until just before the end of the competition (once it was announced). Knowing the timeline would've allowed for more varied prototyping/exploration.

B. Next Steps

Although the challenge has concluded, the MITRE team will continue work on supporting contact tracing via mobile devices. The next steps being considered by the team include:

- More data augmentation/noise to attempt to prevent the network from memorizing values
- Implementing callback to prevent overfitting
- Collect and process more data, especially in a more natural manner. This could potentially provide natural variability and thereby increase the potential accuracy of the model. We would also like to do the data collection with multiple devices and device types in various environments (indoor and outdoor).
- A more robust review of model layout options.

REFERENCES

[1] https://www.nist.gov/system/files/documents/2020/07/01/2020_NIST_Pilot_TC4TL_Challenge_Evaluation_Plan_v1p3.pdf

[2] https://github.com/mkrangle/MITRE-Range-Angle-Structured

[3] M. Qathrady, A. Helmy,. 2017. Improving BLE Distance Estimation and Classification Using TX Power and Machine Learning: A Comparative Analysis, MSWiM '17, November 21–25, 2017, Miami, FL, USA

[4] https://mlflow.org/

© 2020 THE MITRE CORPORATION. ALL RIGHTS RESERVED