2019-08-01

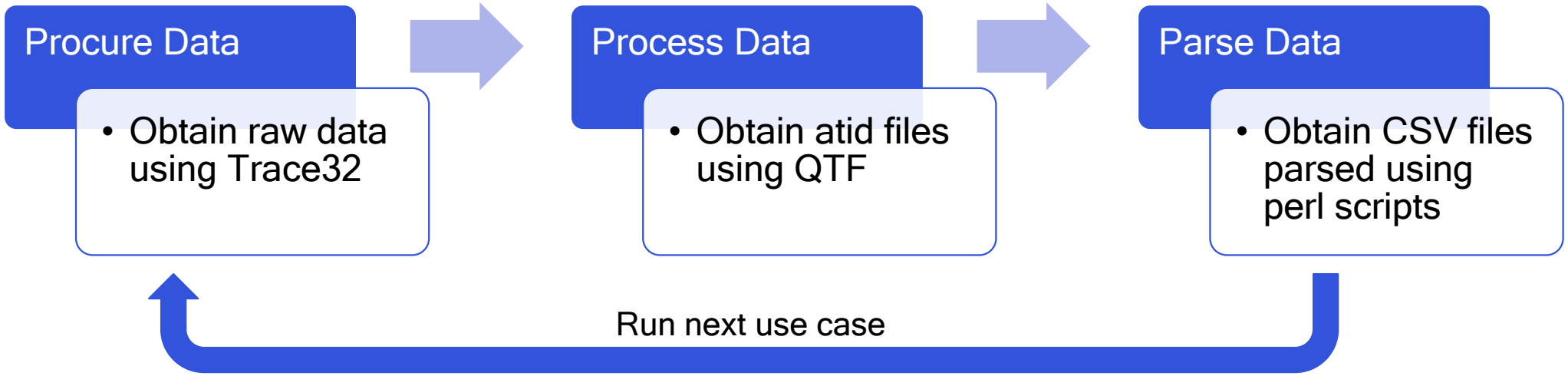
Qualcomm

QoS Optimization with ML

QC-Mobile (SPT)

Automation Framework

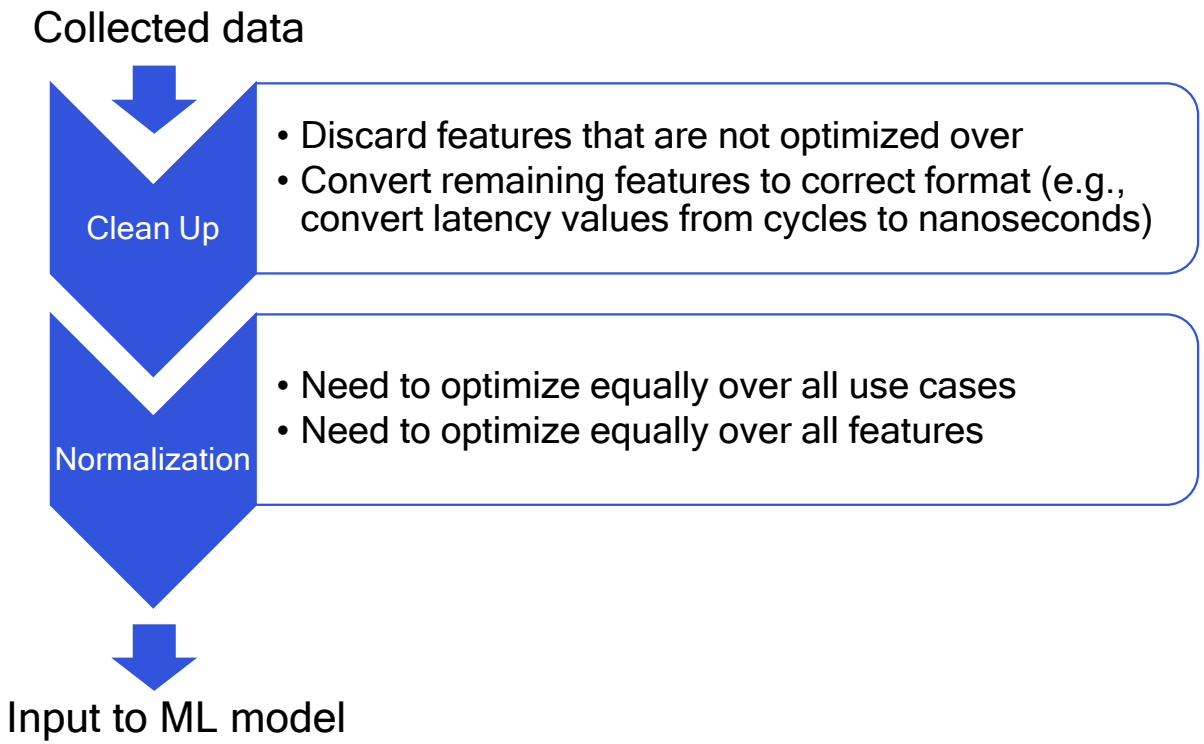
- Upgraded the existing automation framework used to automate the task of running use cases and collecting training data for the ML model.

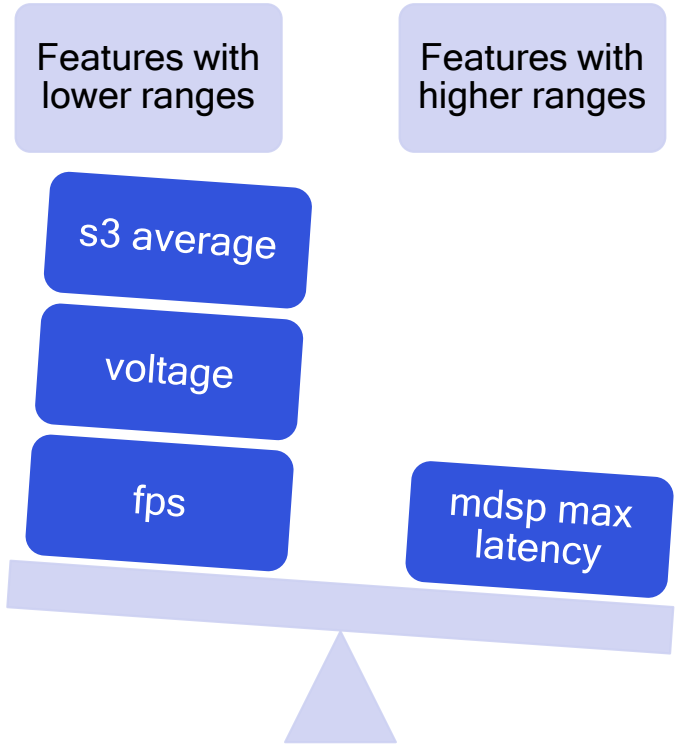


- The above automation cycle is used for training data obtained using bus profiling and other data (e.g., fps) is obtained directly from the 'Procure Data' step of the automation cycle.
- The use cases, along with the number of runs for each use case, can be specified at the start of the automation script. Furthermore, the CMM scripts for the bus profiling and its accompanying perl parser scripts can also be specified at the beginning of the automation script.

Clean Up and Normalization

- Once all the data is collected and stored at a designated location, the automation script then postprocesses this data to convert it into a suitable format for the ML model.

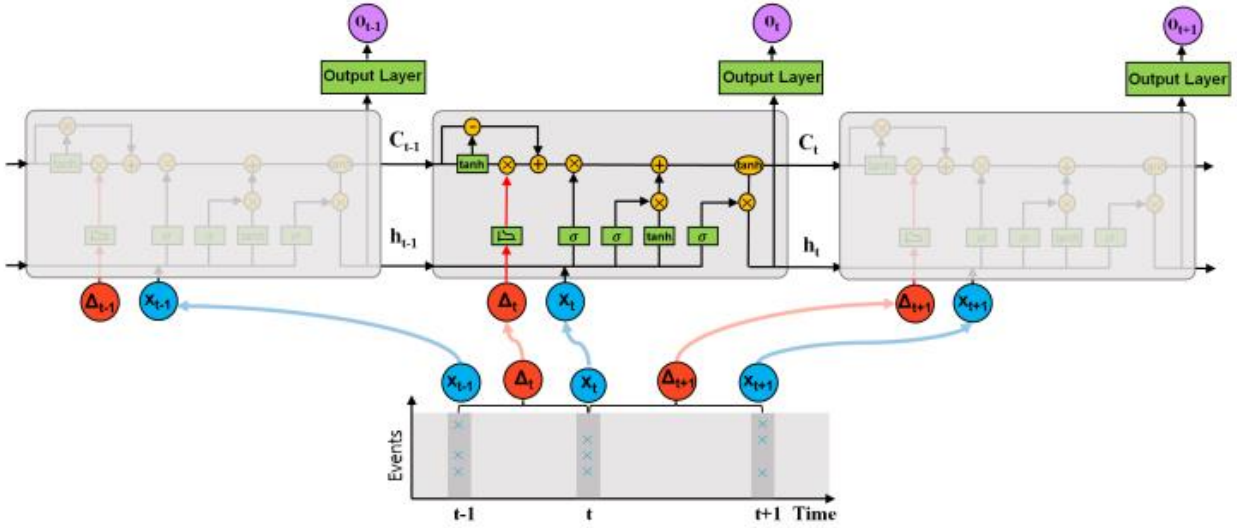




- Use cases with features possessing higher ranges are given a higher priority and Normalization fixes this issue by subtracting the mean and dividing over the standard deviation for each of the features that are optimized over.

LSTM Units

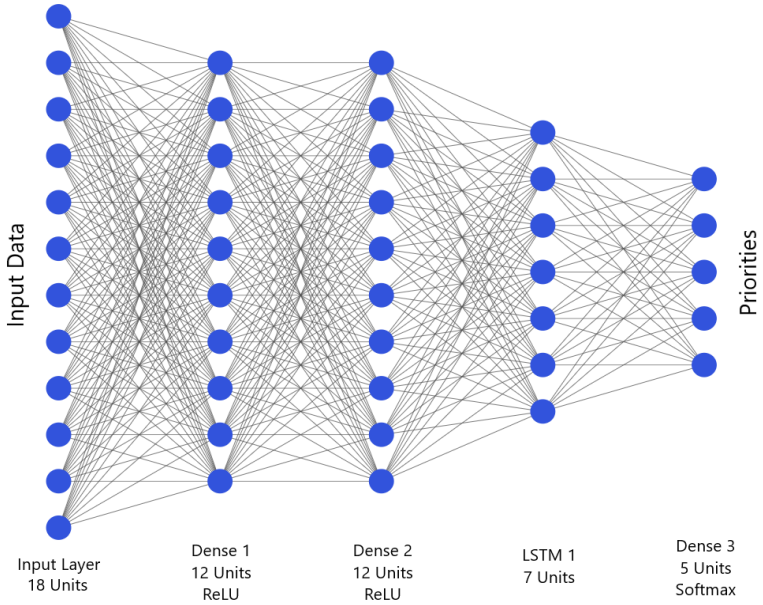
- Long Short-Term Memory (LSTM) units are a special type of neural network that contain loops, thus allowing the LSTMs to retain information in the network. This grants LSTMs the capability of possessing memory, which is extremely useful for capturing trends in time-domain data.

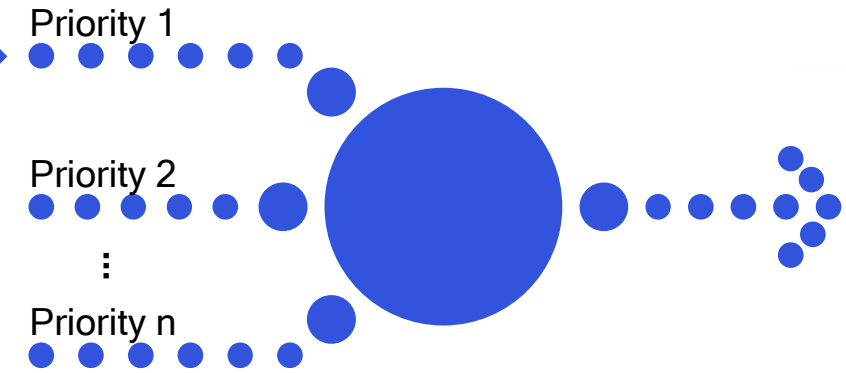


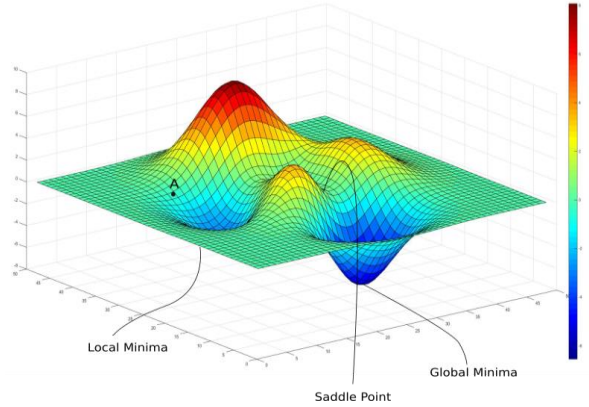
- We attempted other approaches including dense units to create a Multi-Layer Perceptron (MLP) model and 1-D convolution to create a Convolutional Neural Network (CNN) but to no avail as they yielded test accuracies of 60-70%, which is not nearly good enough for the task at hand.
- We finally resorted to using LSTMs to exploit temporal relations in the time-domain data we collected to incorporate memory into the predictions of priorities.

ML Architecture

- In order to determine the best possible architecture that was suitable for predicting the priorities, we used Neural Architecture Search (NAS), which allowed us to create an ideal architecture while preventing both underfitting and overfitting of the training data.







- We chose to implement the ML model as an ensemble of classifiers rather than a regression model to avoid predictions non-existent priorities.
- The ML model performs gradient descent over the parameters to be optimized and has a test accuracy approximately 92% after training for 10 epochs.