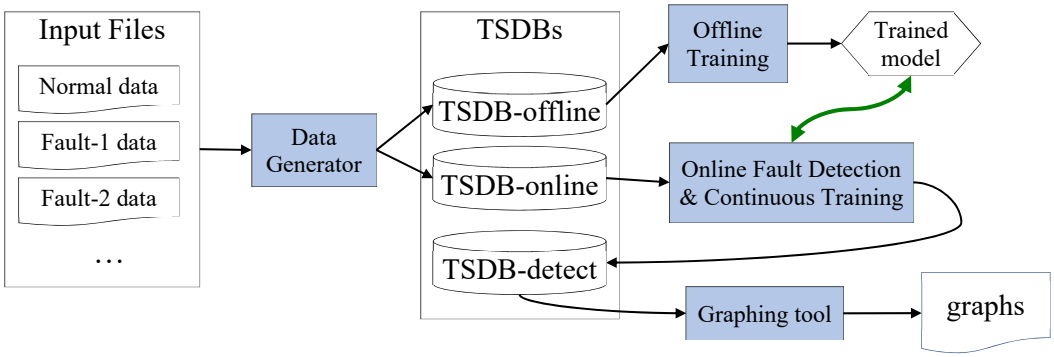
IoT Course Project 2

High level sketch

- Project concept
  - IoT is widely used in industry for problem analysis and fault diagnosis
  - Sensors are placed around to get input readings
    - Could be from machines, the environment, or detecting activities
    - Sensors could be hardware sensors or software sensors (e.g., detecting the cpu utilization, or detecting what activities is being performed on a computer
  - Classifiers are trained to recognize normal and failure conditions from sensor inputs
  - Sensor data may have trend change
    - For example, environment temperature change may impact sensor readings
    - For example, a system getting worn out may have different normal sensor readings compared to when the system is new
    - So, input sensor data should not only be used for failure detection, but also for continuous training
- Offline training
  - Assume that we already have some data for recognizing normal and failure cases
  - Feed the data with the labels to neural network to learn to recognize failures
    - We will use Google tensor flow
  - Let the trained model be M1
- Online failure detection and continuous training
  - Data continuously got collected from the sensors, we need to
    - (1) Apply the existing model obtained earlier (M1) to D1 to determine whether the system is healthy or has failures
    - (2) Assume that some expert will determine whether the failure detection results are correct and correct them if they are wrong. To improve the trained model and to support proper learning while data trend shifts, we need to feed newly labeled D1 to the neural network and get new model M2
    - (3) Replace M1 by M2 and apply it to newly collected data D2 in the next time round
  - Perform steps (1)-(3) iteratively
- Data storage
  - New data gets collected continuously, and they should be fed to a storage for upcoming analytics and potential further processing
    - Use a TSDB for the storage
    - We will use InfluxDB as our TSDB since it is more stable



Implementation

- Data Generation
  - Download bearing data
    - http://data-acoustics.com/measurements/bearing-faults/bearing-5/
  - Merge the normal and faulty data randomly (by window) to simulate a monitoring data stream
    - Consider a window size MW (merge window) for data merging, and MW should be large enough so that the data will not be overly split
    - Select many normal data windows and insert some fault data windows
    - Allow the data to repeat to create infinite monitoring data flow
  - Write a Data Stream Simulator
    - Read from the generated dataset into a buffer, write the buffer to TSDB at a fixed rate
      - TSDB generally accepts data segment by segment, i.e., monitoring system does not send data one entry at a time, but collect data in the buffer and send it when buffer is full, to avoid high communication overhead
      - Let buffer window size be BW, BW < SW
      - E.g., BW = 10 milliseconds
    - About half of the real data (selected segments) can be written to TSDB-offline for offline training
      - You need to add label to the offline training data
    - The entire real dataset will be used repeatedly to feed into TSDB-online, at a fixed rate, for online fault detection and continuously training
      - When you feed the data to TSDB-online, they should not have the label
    - You need to properly adjust the TSDB parameters
- Offline Training
  - Read the data from TSDB-offline and feed them to the learning system
    - Use the method in the paper: Gear Fault Diagnosis Using Discrete Wavelet Transform and Deep Neural Networks
    - You can choose to skip the wavelet transform, which is not the focus of this project, but it would be nice if you do include that part
    - You will need to divide the data to feed to the neural network
      - Let the window size be TW (training window), TW < BW < SW and it is better that SW is multiple of BW and BW is multiple of (or the same as) TW
  - Obtain the trained model M1 and pass it to the Online unit
- Online Fault Detection and Training
  - Read the data from TSDB-online periodically (with size PW, processing window) and process these newly collected data Dcur
    - Need to setup a sync mechanism to make sure that the data Dcur is fully available
      - Explore what the TSDB has offered for easy sync

        - The continuous query can be used for this purpose (or any other method)
        - These data should not be labeled
        - Dcur should be divided into TW to feed to neural network
        - PW can be the same as BW
    - Apply the most current Trained Model Mcur to determine whether there is a fault and if so, the fault type
        - Write the fault detection results to the TSDB
    - Add label to the data (the original fault label, not the label from your detection result)
    - Feed the labeled data to the neural network to perform continuous training and modify Mcur into the next model Mnext
- Online fault detection result presentation
    - Use the graphing utilities provided by the TSDB to graph each measurement
    - Fault data graphing (accurate labels)
        - Your online data, after being labeled, can be fed back to TSDB
        - Set normal = 0, different fault types as different positive integers
    - Fault data graphing (labels of the fault detection results)
        - Graphing the data obtained from your online fault detection (in TSDB)
        - Use the same fault value settings as above
        - Layout the two fault data series next to each other and in different colors to allow visual comparisons
    - Fault diagnosis result reporting
        - Output the fault diagnosis analysis accuracy in various metrics in a text window
    - You can explore different parameters and report these results