Wireless Sensor Network Based Advisory System for Apple Scab Prevention – M.Tech. Thesis by Kriti Bhargava

This paper served as a good starting point for me to explore WSN’s. Many aspects of the design and implementation were covered, including real-world constraints, network architecture, communication protocols, sensor algorithms, etc. My biggest takeaways are:

* Transitioning from an ideal environment to a real-world implementation requires a lot of planning:
  + Sensor topology: How the sensors will communicate, like hierarchy, P2P
  + Location topology of where the sensors will be deployed
  + Varying “normal” conditions of various locations
* We can use simulations to test the network performance and transmitted data
* There is a delicate balance between data availability and sensor battery life
  + Communication drains battery the most, so constant data transmission would drain the battery very quickly

The code and implementation were hard to understand at times because some of it was in a language I presume to be specialized for the sensor (Section titled MIC for Apple Scab Module). It was nice to see C!

A Fog Computing Approach for Localization in WSN – Kriti Bhargava, Stepan Ivanov

The transition from the thesis to this paper is similar to a shift from a centralized computing approach, where sensors simply send data to the cloud or machine which does computations based on that, to shifting some of the computation to the sensors themselves. The objective is to perform operations on that data which allow vital information to be condensed and thus reduce the power spent on communication. The power spent on extra computation must be less than the power saved on communication for this to be a viable strategy.

I was exposed to edge mining tools and related algorithms. These were covered very briefly as this was a short paper so this paper assumes some understanding of these. The graphics which showed the testing results were very useful for me to understand how to gauge performance. I wonder what the challenges might be in implementing the AAL. One could be variations in body movement and position for different users when doing the same activity which makes it difficult to classify accurately for each user.

A review on genetic algorithm: past, present, and future - Sourabh Katoch, Sumit Singh Chauhan, Vijay Kumar

I came upon this paper when I was looking for a more in-depth look into genetic algorithms, which were introduced in the Fog Computing paper. I focused on Section 3, which provides an overview of genetic algorithms. It was really cool to see how it simulates the combination of randomness and improvement in evolution. The formation of fitness criteria seems like an interesting challenge because genetic algorithms attempt to optimize something but we don’t necessarily know what an optimal result would look like.

Edge Mining the Internet of Things - Elena I. Gaura, James Brusey, Michael Allen, Ross Wilkins, Dan Goldsmith, and Ramona Rednic

This paper provided more detail on how edge mining algorithms (SIP, ClassAct, BN) work. The most valuable part to me was the pseudo code because I knew precisely what the algorithms would do. The primary method of evaluation was message reduction because that is the reason why edge mining is useful. All methods use a definition of an event: a data point which cannot be predicted using the given methods. The predictions change overtime based on previous data, meaning that the sensors can continuously learn.

Leveraging Fog Analytics for Context-Aware Sensing in Cooperative Wireless Sensor Networks – Kriti Bhargava, Stepan Ivanov, Diarmuid McSweeney and William Donnelly

This paper served as a transition from regression-based SIP algorithms into classification problems. The classification problem in this case was the activity of cows. There were many instructive pieces in this paper:

* There are often data points which overlap among classes, leading to some amount of error due to the indistinguishability of measurements.
* Sensors can “work together” by communicating their respective states, further improving the accuracy of the classifications. Nodes of closer proximity are obviously more relevant to the classifications. However, there is a trade-off because having more nodes collaborate requires many more signals to be sent -> greater power usage
* Training the decision tree model:
  + Analysis of data is performed to determine which measurements are most useful for differentiating between states. In this paper, the axes of the sensor accelerometers were compared.
  + The window of previous measurements to include and weights of previous measurements should be compared.
  + Epsilon trade-off: lower epsilon means more sensitivity, but this could lead to false positives of state change. Finding the right epsilon is important.

Implementing a decision tree classifier on an Arduino: <https://eloquentarduino.github.io/2020/10/decision-tree-random-forest-and-xgboost-on-arduino/>

This blog post introduces decision trees and random forests as tools for classification. It details how to use Python libraries to generate decision trees based on training data, validate the model and then export it to C code for use on an Arduino. The decision tree is exported as a bunch of nested conditional statements. I used this to create the NFL quarterback decision tree which classifies whether a quarterback is in the Hall of Fame based on career statistics.

Feed-forward versus feedback structures: <https://blog.paperspace.com/feed-forward-vs-feedback-neural-networks/>

This blog post provides an introduction to neural network structure:

* Data passes from the input layer to hidden layers and finally the output layer.
* Weights are associated with connections between nodes.
* Activation functions may be used to incorporate non-linearity in the operations between neurons to accommodate for non-linear patterns.

It also explains the difference between feed-forward and feedback neural networks. My biggest takeaways are:

* Feed-forward networks have a one-way flow of information through the layers. The primary mechanism for learning is back-propagation. Back-propagation is adjusting a neural network's weights based on error rates.
* Feedback neural networks can have flow in any direction, which can create feedback loops. It is a more dynamic system which achieves equilibrium after successful training.

Understanding the mathematics of ANNs and back-propagation: <https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd>

To extend upon the explanations in the previous article on feed-forward/feed-back networks, we found an article which covers the fundamental mathematical ideas behind ANNs. It was cool to see how concepts in Linear Algebra and Multivariable Calculus are used.

Implementations:

* Using the Arduino sensors: To start out, we created simple Arduino sketches which read values from the Gyroscope, Humidity and Temperature sensors and print significant changes in those values. Significance was defined using an arbitrary delta value where if the new reading differed by more than the delta value, then the change is significant.
* Doubly Exponentially Weighted Moving Average: The previous method of determining a significant change is very basic and requires a lot of communication from the Arduino. We used the same sensors and changed our prediction model so that instead of only using the previously value, we use a formula which incorporates previous values. The goal is to reduce the amount of significant changes recorded while still maintaining the approximate shape of trends.
* Decision Trees: To practice the implementation of a decision tree, we created a model for classifying NFL quarterbacks as Hall of Fame inductees based on career statistics. We used Python to create, train and validate the model, then export it to C. When the Arduino receives statistics, it outputs a true/false representing Hall of Fame / not Hall of Fame.
* ANNs: We explored examples of using the TinyML library to generate ANNs. One simple ANN takes the audio of a person speaking and determines whether they are saying “yes” or “no.” Natural language processing is very complex so tasks beyond this can quickly exceed the computational capabilities of the Arduino. The TinyML examples were taken from the library GitHub: <https://github.com/tinyMLx/arduino-library/tree/main>