GA Tech Capstone Analytics Practicum Proposal

A partnership with Sandia National Laboratories

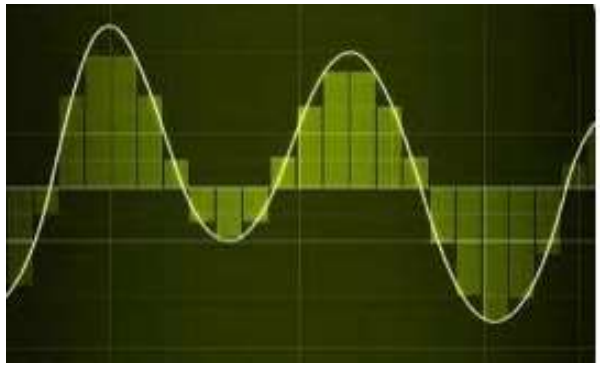
Stephen Smith

Data Analyst & GA Tech MSCS Graduate 2020

(Key Contact for this Practicum)

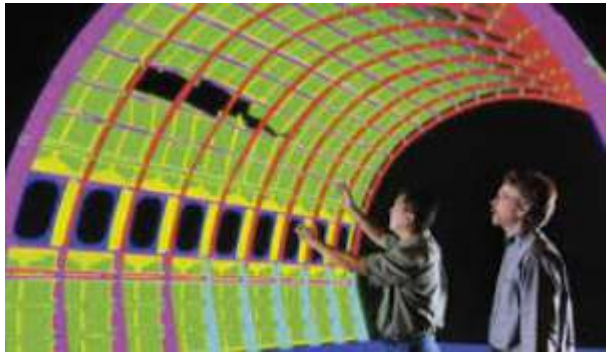
Proposal Topic

Machine Learning & Signal Analysis to automatically predict component degradation



Who We Are and What We Do

- Sandia National Laboratories creates and fosters an environment where some of the most talented engineering and computer science professionals form teams to solve some of the nation's most challenging engineering programs. Our national laboratory been solving the most difficult technical challenges for over 75 years, serving in the national interest.
- A team of Sandia engineers and computer scientists met to propose a challenging technical Capstone Analytics Practicum project containing several real-world applications



Background for Proposal Idea

- As hardware components age over time, the critical signals those components generate can change
- Physically testing critical components is expensive and time-consuming
- We would like to introduce you to an innovative way components could be tested automatically by checking the signal outputs they produce and comparing them to a dataset of “healthy” signals by using **Machine Learning**.

General Proposed Machine Learning Process

1. Collect a dataset of degraded signals and signals that have not degraded
2. Extract relevant information from those signals that help identify the degradation (this could include statistical measurements like mean, variance, and skewness
3. Split the dataset into training, validation and testing sets
4. Train the model
5. Evaluate the model
6. Use the model

Using Python and Toolboxes to Work Toward a Solution

- Scikit-learn
- Tensor Flow
- PyTorch
- Keras
- NumPy & Pandas
- Matplotlib & Seaborn

Generating Your Own Artificial Signals with Python

- No dataset will need to be provided for this study. Here are techniques you can use to conduct this work:
  - Have your team think of a type of signal you're interested in, like a noise signal for instance
  - Use Numpy to add random noise to the signal
  - Use filtering in signal processing to degrade a signal like high and low pass filters
  - Then use built in functions like short-time Fourier transform or a wavelet transform to generate a degraded signal (you won't have to know how these transforms work, just how to work with them)
  - Do no transforms to your original signals
  - Collect a robust dataset of strong and degraded Signals

Do Machine Learning and Test the Model

- Perform Machine Learning to classify strong and weak signals
- Test your model out to see if your model can detect bad and strong signals you introduce.
- Demonstrate your results in a presentation and final report
- Capture the procedure you used to conduct your artificial machine learning signal classification and show how it could be applied to real world signals
- Pat yourself on the back, because what you have developed has big real-world implications.

This experience is extremely valuable in industry.

Backup Slide – Full Description of the Process

1. **Collect and preprocess the data:** Collect a dataset of signals that have degraded and a dataset of signals that have not degraded. Preprocess the data by cleaning, normalizing, and transforming it into a format suitable for machine learning.
2. **Feature extraction:** Extract relevant features from the signals that can help identify degradation. These features could include statistical measures such as mean, variance, and skewness (measurement of distortion of symmetrical distribution or symmetry in a data set), as well as spectral features such as frequency components or spectral power.
3. **Split the data:** Split the dataset into training, validation, and testing sets. The training set is used to train the machine learning model, the validation set is used to tune hyperparameters, and the testing set is used to evaluate the model's performance.
4. **Train the model:** Train a machine learning model using the training set. The model should take the extracted features as input and output a prediction of whether the signal has degraded or not.
5. **Evaluate the model:** Evaluate the performance of the model using the validation and testing sets. Metrics such as accuracy, precision, recall, and F1-score can be used to assess the model's performance.
6. **Use the model:** Once the model has been trained and evaluated, it can be used to make predictions on new signals. If the model predicts that a signal has degraded, it may be necessary to take corrective action to restore the signal's quality.

Backup Slide – Python Toolbox Descriptions

1. **scikit-learn:** This is a widely used Python library for machine learning that provides a wide range of algorithms for classification, regression, clustering, and other tasks. It also includes tools for model selection, data preprocessing, and feature engineering.
2. **TensorFlow:** This is a popular open-source machine learning library developed by Google. It allows users to define, train, and run machine learning models using Python. TensorFlow provides a wide range of tools for building and training neural networks, as well as tools for working with large datasets.
3. **PyTorch:** This is another popular open-source machine learning library that provides a dynamic computation graph and allows for more flexible model architecture. It is particularly useful for rapid prototyping and development of complex models.
4. **Keras:** This is a high-level neural networks API that can run on top of TensorFlow, Theano, or PyTorch. It allows users to quickly build and experiment with neural network models using Python.
5. **NumPy and Pandas:** These libraries provide powerful tools for data manipulation and analysis, and are often used as a foundation for machine learning analysis. NumPy provides support for large, multi-dimensional arrays and matrices, while Pandas provides data structures and functions for working with tabular data.
6. **Matplotlib and Seaborn:** These libraries provide popular visualization tools for data analysis and machine learning. Matplotlib provides a wide range of visualization options, while Seaborn provides a high-level interface for creating informative and attractive visualizations.