It is well known that two-dimensional convolutional neural networks are powerful image recognition machine learning techniques. There is another less known convolutional neural network called the one-dimensional convolutional neural network (1D CNN) which has many applications. Some of the applications of 1D CNN are time series forecasting using regression and classification use cases such as faulty component detection in manufacturing machinery, natural language processing (NLP), human activity monitoring, patient specific ECG classification, structural health monitoring and anomaly detection in power electronic circuitry. (Kiranyaz, 2019).

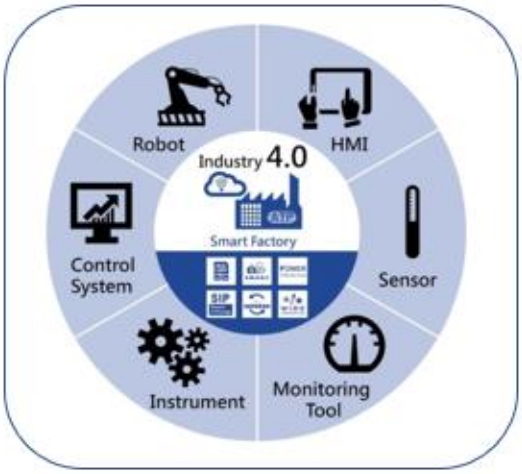
In this paper, the focus will be classification of faulty components using sensor data in manufacturing machinery. Feature engineering is the heart of the classic approach to signal processing and machine learning using features extracted from the output of the Fast Fourier Transform and Discrete Wavelet Transform for this paper. The contemporary approach is to use 1D CNN and raw signals as input to the 1D CNN model. Both techniques will be used in this paper and the results compared.

Table of Contents

I. Executive Overview

Manufacturing is undergoing a major change with Industry 4.0 which refers to the fourth industrial revolution. Industry 4.0 is a trend toward automation and data exchange in manufacturing technologies and processes which includes the cyber-physical system (CPS), the internet of things (IoT), industrial internet of things (IIOT), cloud computing and artificial intelligence (Wikipedia, 2019) which enables the creation of smart factories. In Industry 4.0, interconnected sensor-enabled equipment is required in smart factories and continuously generate data for every aspect of the manufacturing process; data that can be collected and analyzed in real time (IVEDIX, 2017).

Machine Learning is a key component for the predictability of failures of equipment using both raw sensor data and engineered features from raw sensor data in manufacturing. Thus, providing foresight to predictive maintenance procedures before an unscheduled outage occurs. The promise of Industry 4.0 is the enhancement of manufacturing productivity which go beyond today's gains. As the following diagram shows, sensors will play a major role in the push towards manufacturing automation and the smart factory. This project uses sensor data collected from monitoring ball bearing assemblies.

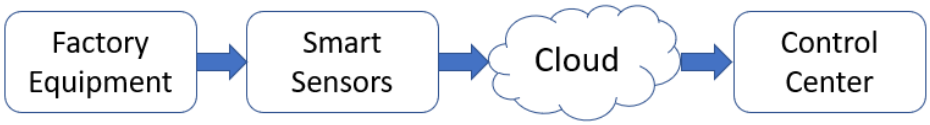


Source: (Goel, 2017)

Two different approaches for predicting faulty bearings were used. Compared are classic methods of predicting faulty bearings using engineered features and contemporary methods using one-dimensional convolutional neural networks (1D CNN). Results showed that both the classic and contemporary methods performed well in predicting faulty bearings. The main difference in the two methods is an understanding of domain signal theory is required for classic methods while very little knowledge of signal theory is needed for 1D CNN. However, understanding the results and improving results based on the data is much easier with classic methods due to requirements of having domain knowledge. Two- and three-dimensional CNN often required large amounts of data for training; however, 1D CNN required about the same amount of data as classic methods. This makes it very attractive as an alternative to the classic methods.

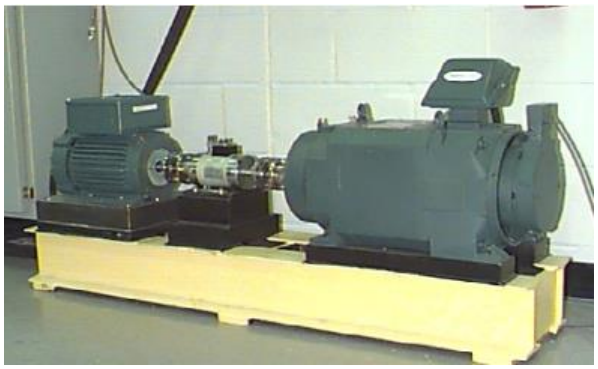
II. Introduction

In a smart factory, the following depicts the roles of sensors in the smart factory. Sensors that monitor vibration, temperature, acceleration, audio, images and other signals are used.



Sensor data is more available than ever. Industry 4.0 refers to the next step in industrial technology, with robotics, computers and equipment becoming connected to the Internet of Things (IoT) and enhanced by machine learning algorithms. With this accessibility, managers, executives and data scientists can use that insight to improve the efficiency and productivity of the whole operation. People are now looking at how to leverage industrial IoT sensor data to project things that may happen such as predictive maintenance, line management or quality control (Tower-Clark, 2019).

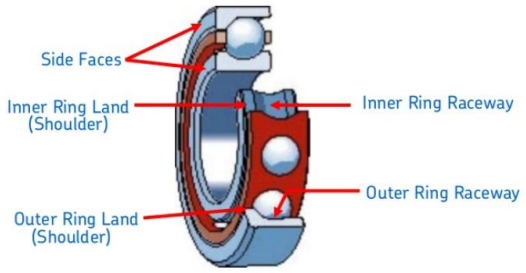
This project uses sensor accelerometer data to predict failures of industrial equipment. Data was created by the Case Western Reserve University containing normal and faulty bearings. Experiments were conducted using a 2 hp Reliance Electric motor, and acceleration data was measured at locations near to and remote from the motor bearings using sensors. The following picture shows the machinery.



Source: (CaseWestern)

Accelerometers were placed on the fan end and drive end of the motor as well as the base part of the motor. Accelerometers were placed at the 12 o'clock position at both the drive end and fan end of the motor housing. Outer raceway faults are stationary faults; therefore, placement of the fault relative to the load zone of the bearing has a direct impact on the vibration response of the motor/bearing system. In order to quantify this effect, experiments were conducted for both fan and drive end bearings with outer raceway faults located at 3 o'clock (directly in the load zone), at 6 o'clock (orthogonal to the load zone), and at 12 o'clock. Vibration signals were collected using a 16 channel DAT recorder, and were post processed in a Matlab format (CaseWestern). Vibration data was collected for both inner and outer raceways for both drive end and fan end parts of the motor.

The following figure shows the components of a SKF bearing assembly.



Source: (Ahamed, 2015)

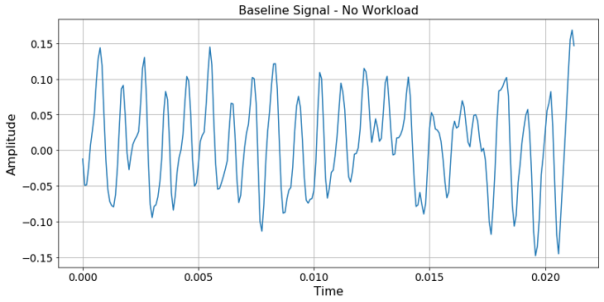
This project only used drive end, inner raceway sensor data. Data was collected under normal operation with workloads of 0 HP, 1 HP 2 HP and 3HP applied to the system. Faulty bearings were introduced using a process called electro-discharge machining (ESD). Defective bearings were introduced with .007 inches, .014 inches, .021 inches and .028 inches of defect using ESD and the four workloads were applied to the faulty bearings as well. There are 4 classes of normal operation and 16 classes of faulty bearing operation which makes a total of 20 different classifications of signal data.

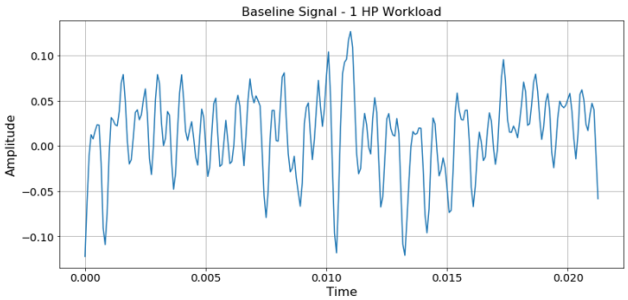
III. Data Acquisition and Exploratory Data Analysis

Sensor data was collected at a rate of 12,000 samples per second and stored in multiple matlab formatted files. Python was used to concatenate all data files into one large array then the data was segmented into 256 samples called a segment. Each segment was then classified as belonging to one of twenty different classes as the following chart shows. This data is for inner raceway, drive end sensor data.

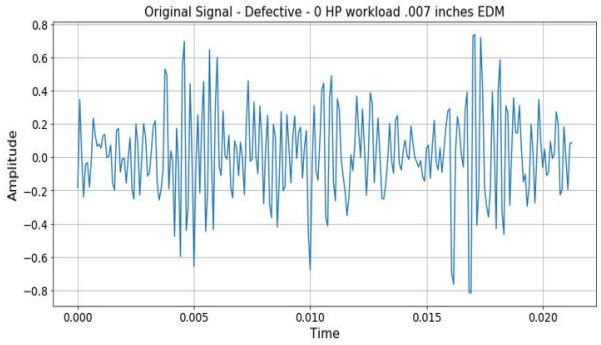
Label	Segments	Data Type	Data Description	File Name
0	952	Baseline Data	0 HP workload normal	97.mat
1	1890	Baseline Data	1 HP workload normal	98.mat
2	1894	Baseline Data	2 HP workload normal	99.mat
3	1897	Baseline Data	3 HP workload normal	100.mat
4	473	Faulty Data	0 HP workload .007 inches EDM	105.mat
5	475	Faulty Data	0 HP workload .014 inches EDM	169.mat
6	477	Faulty Data	0 HP workload .021 inches EDM	209.mat
7	471	Faulty Data	0 HP workload .028 inches EDM	3001.mat
8	476	Faulty Data	1 HP workload .007 inches EDM	106.mat
9	475	Faulty Data	1 HP workload .014 inches EDM	170.mat
10	474	Faulty Data	1 HP workload .021 inches EDM	210.mat
11	474	Faulty Data	1 HP workload .028 inches EDM	3002.mat
12	477	Faulty Data	2 HP workload .007 inches EDM	107.mat
13	475	Faulty Data	2 HP workload .014 inches EDM	171.mat
14	475	Faulty Data	2 HP workload .021 inches EDM	211.mat
15	474	Faulty Data	2 HP workload .028 inches EDM	3003.mat
16	480	Faulty Data	3 HP workload .007 inches EDM	108.mat
17	475	Faulty Data	3 HP workload .014 inches EDM	172.mat
18	476	Faulty Data	3 HP workload .021 inches EDM	212.mat
19	474	Faulty Data	3 HP workload .028 inches EDM	3004.mat

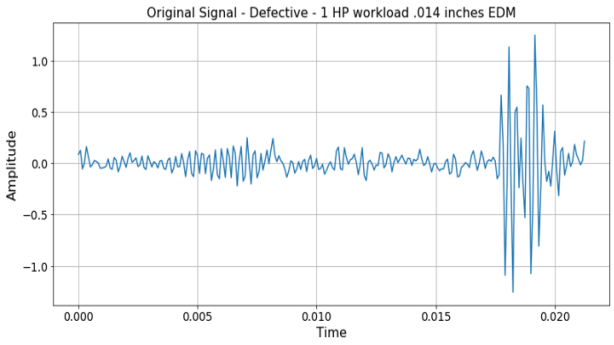
The following shows what typical normal baseline signals of sample size 256 look like.

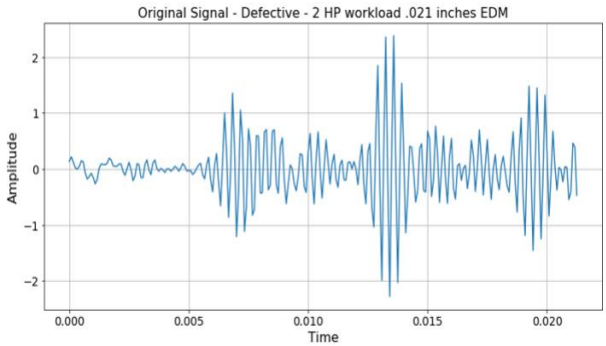


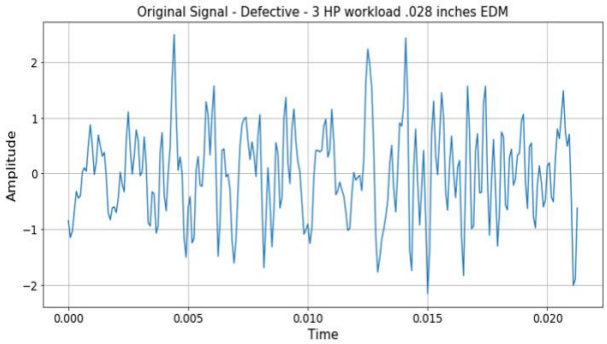


Here are some of the signals of sample size 256 which have faulty bearings represented in sensor data.









There were 14,234 segments created. All of the 14,234 segments in their raw signal form are used as input to the contemporary approach, and the raw signals arwe used as input to the feature extraction processes for the classic approach.

The solution not only needs to determine if it is a normal or defective signal, but also it should classify the workload applied. Normal and defective signals should classify workloads as 0 HP, 1 HP, 2 HP or 3HP. In addition, if it is a defective signal, the defective type (.007, .014, .021, .028) should be determined.

The classic method requires domain knowledge for feature definition and extraction. This approach uses the Fast Fourier Transform and the Discrete Wavelet Transform to extract features through a feature engineering process.

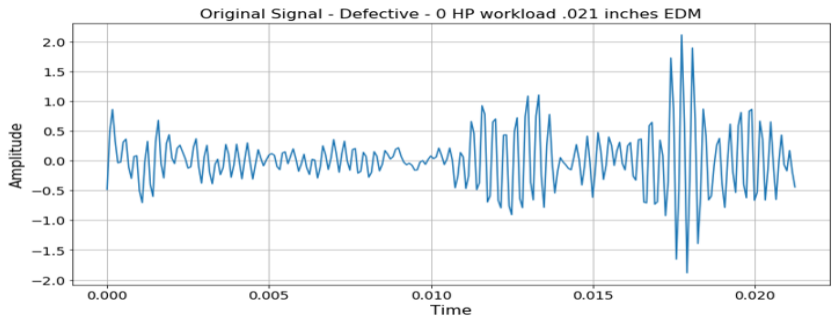
IV. Fast Fourier Transform Feature Extraction

The Fast Fourier Transform (FFT) is an implementation of the Discrete Fourier Transform (DFT). FFT converts the time domain samples to a frequency domain. The time domain is lost in the process. Each of the 256 sized segments are input to FFT. The output of FFT are real and imaginary coefficients for each segment.

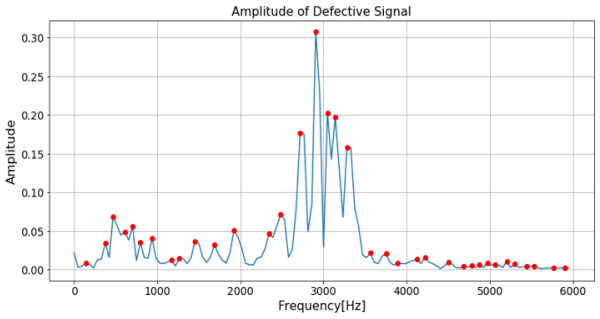
Assume A_k are the real coefficients and B_k are the imaginary coefficients where k=1,2,3, … 128 for each frequency in the frequency domain. Then the amplitude is $\frac{2}{N} * \sqrt{A_k^2 + B_k^2}$ where N is the number of samples (256), and the phase shift is defined as $arctan2\left(\frac{B_k}{A_k}\right)$. The Power Spectral Density (PSD) is defined as $A_k^2 + B_k^2$. In addition, the autocorrelation of the signal is calculated which is the correlation of a signal with a delayed copy of itself.

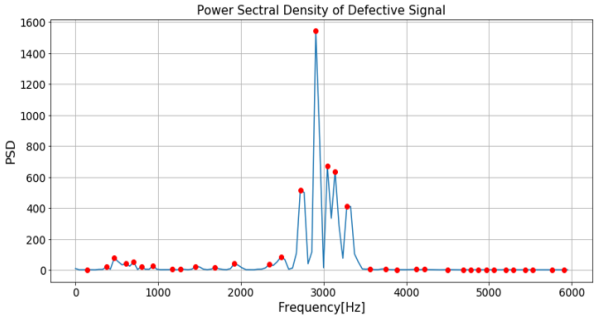
The amplitudes, PSDs and autocorrelations are inputs to the FFT feature extraction process. The FFT feature extraction process simply extracts the peaks of each of the resulting frequency domain values of amplitudes and PSDs. It also extracts the peaks of the autocorrelation wave.

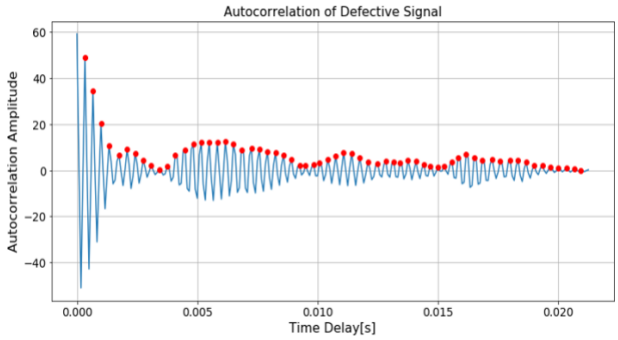
The following is one of the raw signal segments which is input to FFT.



FFT is invoked and the feature extraction process highlights the peaks of each of our extractors. (Taspinar, Machine Learning with Signal Processing Techniques, 2018)







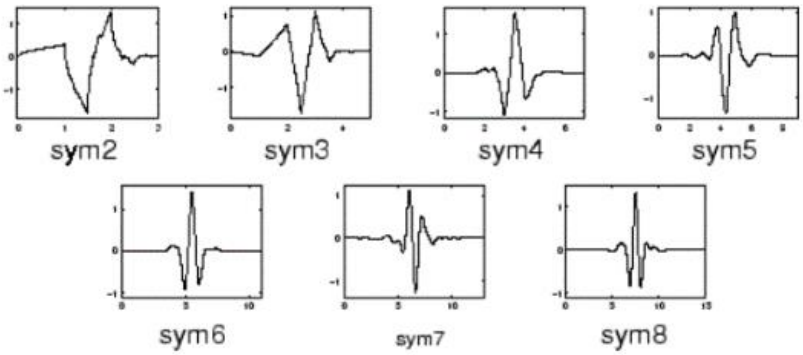
Note that many peaks have been detected. Only the top 10 peaks for each plot are retained as features for classification algorithms. The top n peaks approach is a tuning parameter.

V. Discrete Wavelet Transform Feature Extraction

The Discrete Wavelet Transform (DWT) operates in both the time and frequency domain. This allows for the analysis of a signal in the time and frequency domains. DWT uses concept called a wavelet to analyze signals. A wavelet is a predefined wave which lasts for a small amount of time with mean of zero. There are many families of wavelets. To perform a DWT, you must provide a wavelet as input. Wavelets exist only for a finite duration and come in different shapes. Choosing the right wavelet is important, and the best way choose a wavelet is by trial and error.

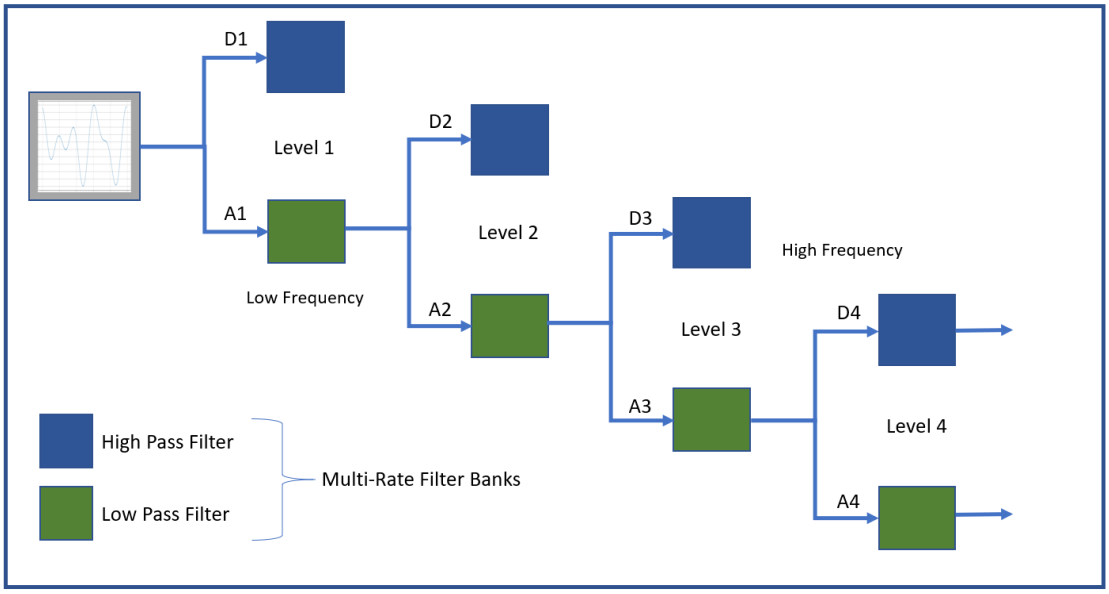
Some wavelet families are haar, db, sym, coif, bior, rbio, dmey, gaus, mexh, morel, cgau, shan, fbsp and cmore.

The following is a set of sym wavelets. They are shown here because it was found that the “sym2” wavelet performed the best for the bearing sensor data.



There are two fundamental operations of the DWT, scaling and shifting. Scaling is the process of stretching or shrinking the wavelet as it passes over the signal in time. The process of moving the wavelet over the signal in time is called shifting. Signals typically consist of slowly changing waves with abrupt short-term changes to the waves. (Devleker) It is the abrupt changes that are of specific interest to machine learning as it provides a blueprint of the behavior of the signal at that time. To

find these slow and abrupt changes, DWT uses wavelets along with high pass and low pass filter banks. The signal is split into high frequency and low frequency signals at each level of filtering. The following shows how high pass and low pass filtering works.



DWT is often used in noise reduction as it separates the base signal from the noisy signals. The python function pywt.wavedec() returns the final set of coefficients for the low frequency result and all of the high frequency sets of coefficients at each level in the process. This information is then fed into three functions which calculate entropy, statistics and crossings on each of the sets of coefficients. (Taspinar, 2018)

The entropy calculation needs as input the probabilities of the coefficients. Once completed, the entropy calculation for the probabilities of each set of coefficients is as follows:

$$-\sum_{i=1}^{n} P(x_i) * \log(P(x_i))$$

The code is as follows:

```
counter_values = Counter(list_values).most_common()
probabilities = [elem[1]/len(list_values) for elem in counter_values]
entropy=scipy.stats.entropy(probabilities) or -sum(probabilities * np.log(probabilities))
```

The statistical features that are calculated on each set of coefficients returned by pywt.wavedec() are as follows:

```
n5 = np.nanpercentile(list_values, 5)
n25 = np.nanpercentile(list_values, 25)
n75 = np.nanpercentile(list_values, 75)
```

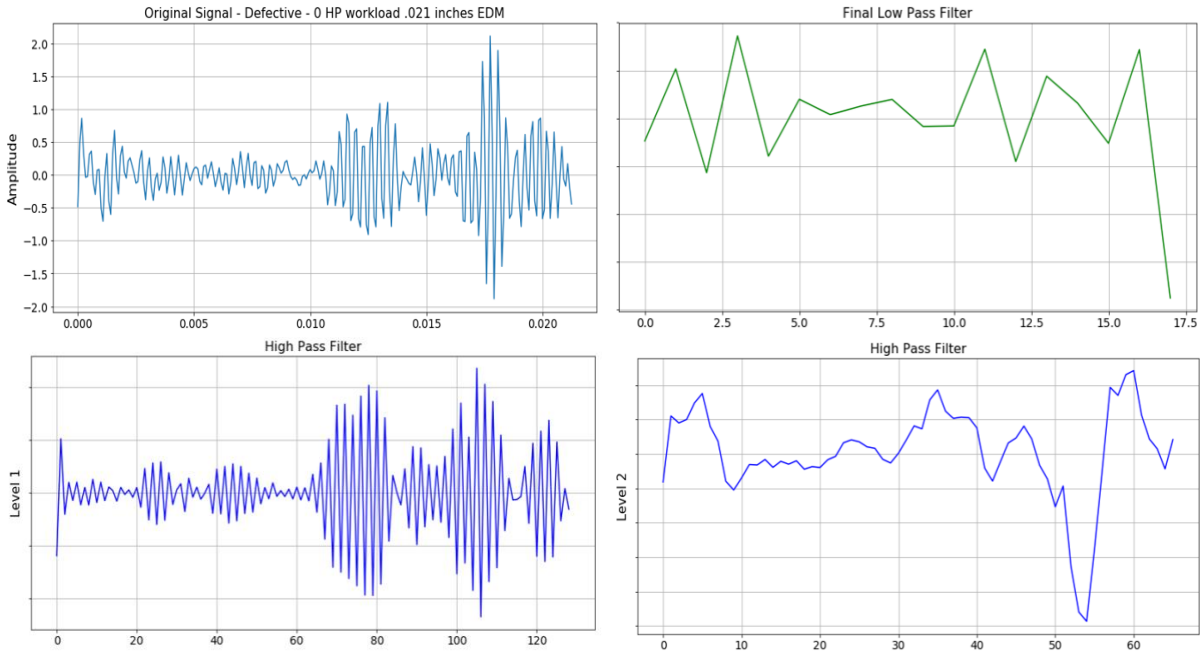
```
n95 = np.nanpercentile(list_values, 95)
median = np.nanpercentile(list_values, 50)
mean = np.nanmean(list_values)
std = np.nanstd(list_values)
var = np.nanvar(list_values)
rms = np.nanmean(np.sqrt(list_values**2))
```

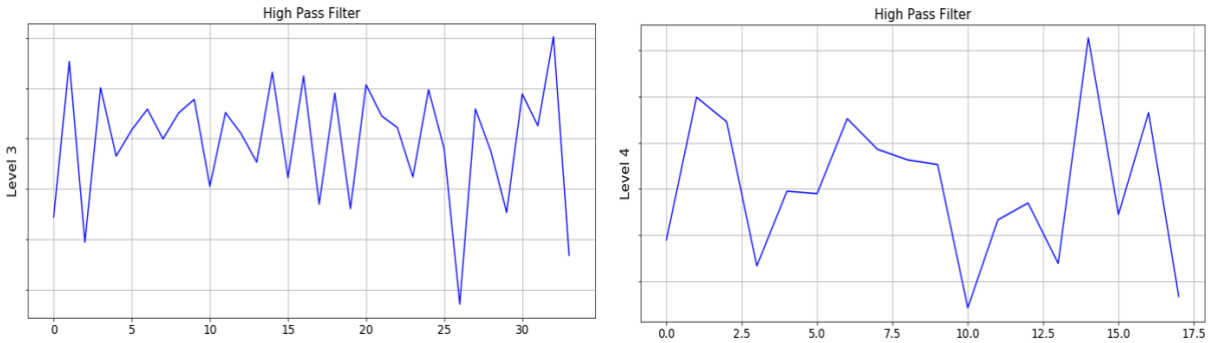
Note: the nan prefix tells the function to ignore nan values.

The final set of features is the number of times the signal crosses the x axis at y = 0 and the number of times the signal crosses the x axis at y = mean(signal_values). The calculation is as follows:

```
zero_crossing_indices = np.nonzero(np.diff(np.array(list_values) > 0))[0]
no_zero_crossings = len(zero_crossing_indices)
mean_crossing_indices = np.nonzero(np.diff(np.array(list_values) > np.nanmean(list_values)))[0]
no_mean_crossings = len(mean_crossing_indices)
print(no_zero_crossings, no_mean_crossings)
```

Here is what the output of wavedec() looks like for one of the signal segments. The initial signal is listed first (the input).



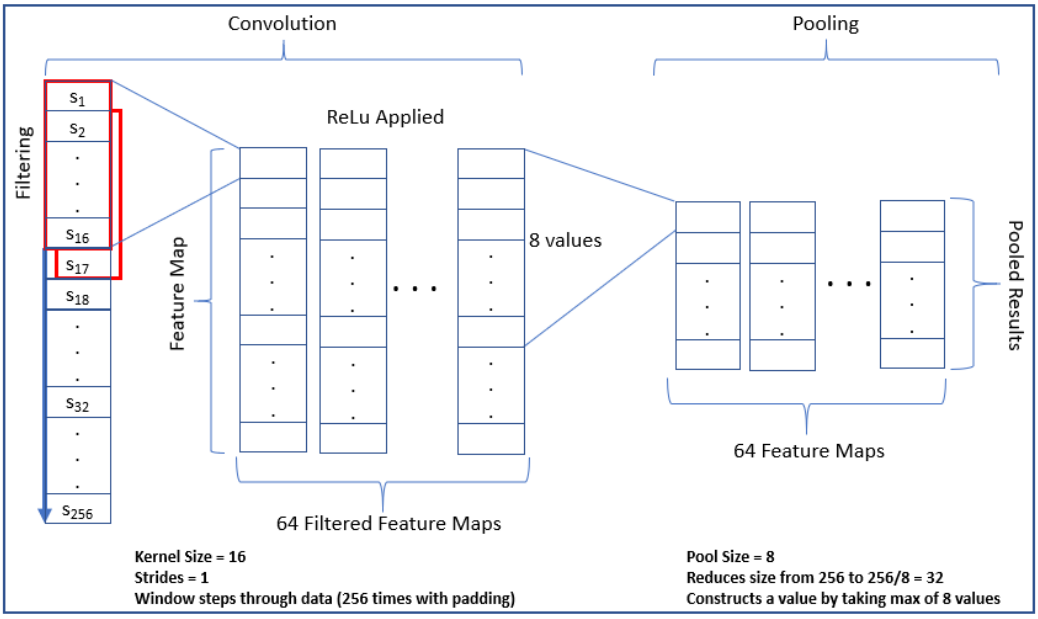


Note the x-axis is halved at each iteration.

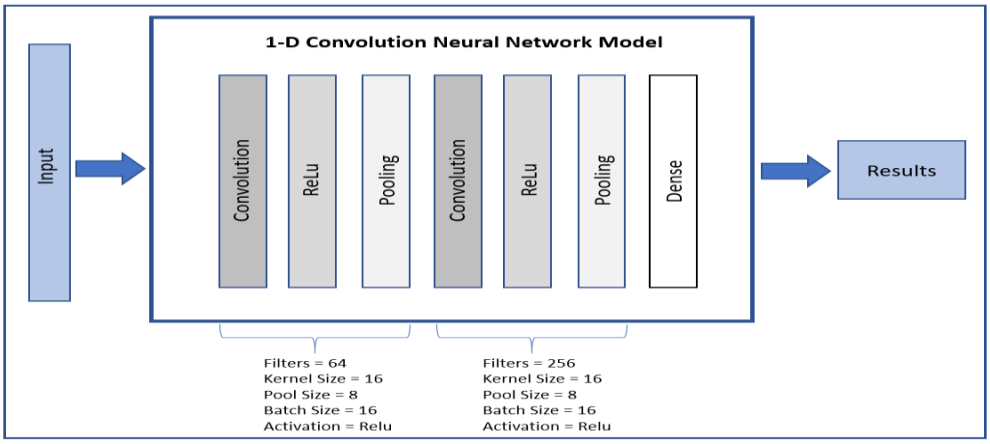
Each set of the coefficients returned by wavedec() is fed into the feature extraction functions and are then combined into one set of features.

VI. 1D Convolutional Neural Networks and Raw Signals

The benefit of 1D CNN is it doesn't need the feature engineering step in which domain knowledge is required. The features are created as part of the execution of the model, and the input to the model are the raw signals segmented into 256 samples each. (Zang, 2017) The following represents the architecture of the 1D CNN used for this project.



Convolution Neural Networks automatically build the features through a process called filtering. The features are tuned through backpropagation. The Convolution/ReLu/Pooling layers can be stacked to create multiple sets of layers. The final set of layers looks as follows:

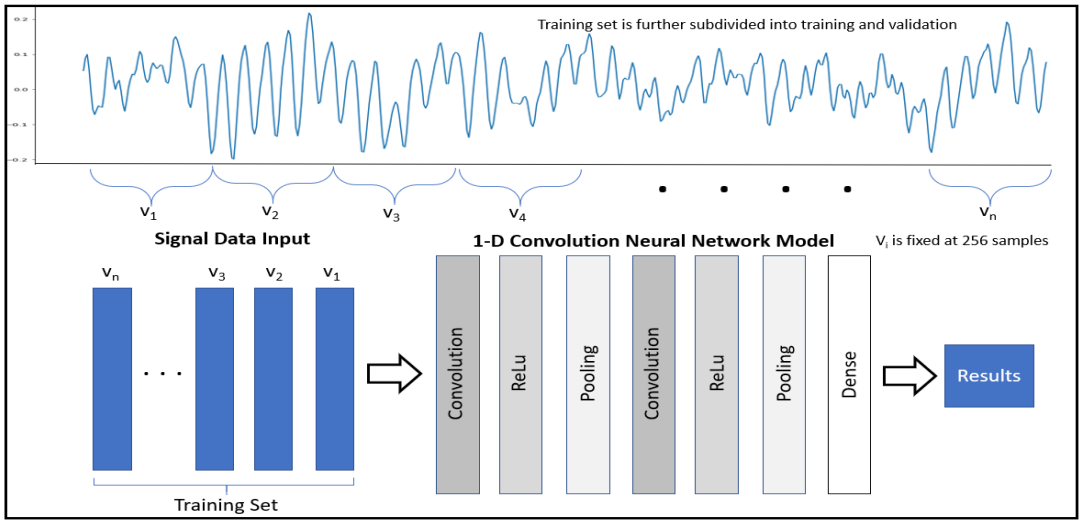


Hyperparameters were tuned as follows:

Segment Size	Kernel Size	Pool Size	Batch Size	Filters	
128	4	2	8	32	64
256	8	4	16	64	128
512	16	8	32	128	256

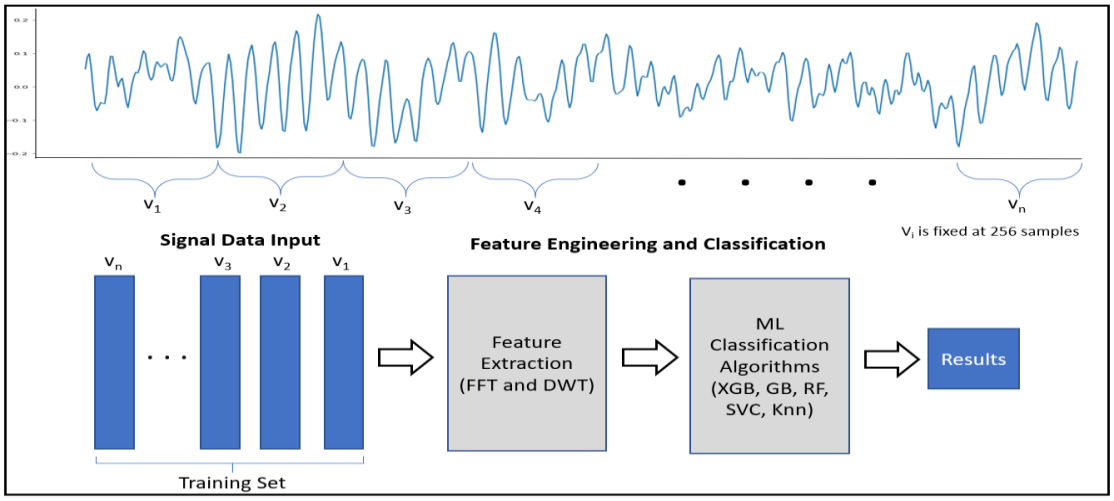
VII. Data Preparation and Execution of Models

The data preparation process for the contemporary approach consists of segmenting the signals into 256 sample segments and performing a train/test split of 70% training and 30% testing. The 70% training was further divided into 70% training and 30% validation. The 1D CNN training process feeds the raw signals directly into the model as follows:



The test set is then applied to the model. Model accuracy, loss and gain plots, classification report and confusion matrix are shown.

The data preparation process for the classic approach is much the same as the contemporary approach. However, the raw signals are not fed into the model directly. A feature engineering layer extracts the features for the classification models as follows:



Each feature set (FFT feature set and DWT feature set) is input to the classification models separately and then the FFT and DWT feature sets are combined and are input to the classification models as one set of features.

Note there is no need to further subdivide the training input as was done in the 1D CNN. Once the features are extracted a series of machine learning algorithms are executed against the model. Testing data is then predicted as usual. Accuracy, classification reports and confusion matrices are created.

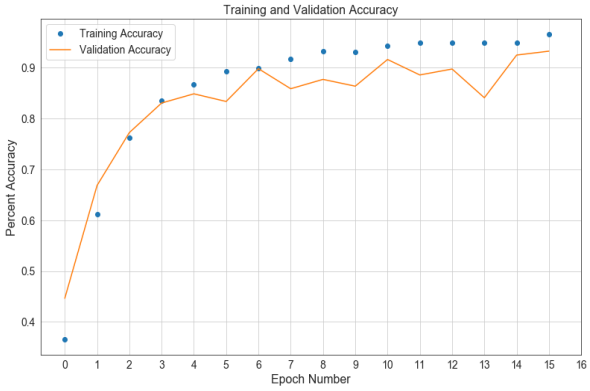
VIII. The Results

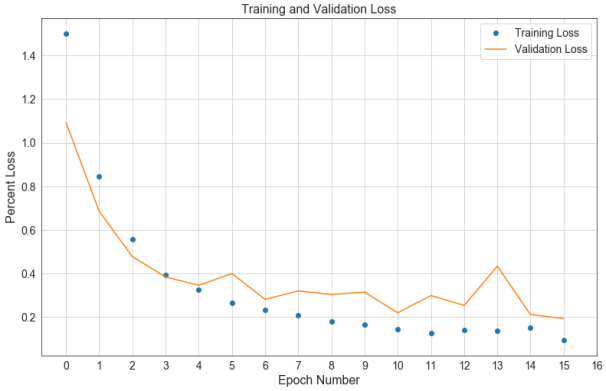
Both the contemporary and classic approaches performed well. The contemporary approach performed the best with an accuracy score of .94. The classic approach had good scores when the FFT features were combined with the DWT features as one set of features. XGBoost and Gradient Boosting performed the best as compared to other classification algorithms using the classic approach. The following is the results of all of the runs.

Approach	Engineered Features Source	Algorithm	Training Accuracy	Testing Accuracy	Execution Time (minutes)
Contemporary	N/A	1D CNN	0.97	0.94	23
Classic	FFT	XGBoost	1.00	0.87	8
Classic	FFT	Gradient Boosting	1.00	0.81	5
Classic	FFT	Random Forests	1.00	0.81	6
Classic	FFT	Knn	0.63	0.54	1
Classic	FFT	SVCLinear	0.70	0.54	1
Classic	DWT	XGBoost	1.00	0.81	13
Classic	DWT	Gradient Boosting	1.00	0.79	4
Classic	DWT	Random Forests	1.00	0.80	7
Classic	DWT	Knn	0.63	0.53	2
Classic	DWT	SVCLinear	0.72	0.43	1
Classic	FFT & DWT	XGBoost	1.00	0.90	17
Classic	FFT & DWT	Gradient Boosting	1.00	0.87	4
Classic	FFT & DWT	Random Forests	1.00	0.86	10
Classic	FFT & DWT	Knn	0.65	0.56	3
Classic	FFT & DWT	SVCLinear	0.83	0.53	1

The two best scores FFT & DWT combined engineered features ran in 17 minutes and 1D CNN ran in 23 minutes. I guess the saying “the besting things in life are worth waiting for” applies here. I only ran the results using CPUs (central processing units). Using GPUs (graphics processing units), the times would have been much lower.

A. 1D CNN Results





Test Loss and Accuracy plus classification report

```
Test Loss: 0.16708979699642926
Test Accuracy: 0.9365488
              precision    recall  f1-score   support

     Class 0       0.98      1.00      0.99       293
     Class 1       0.96      0.99      0.97       581
     Class 2       0.83      0.85      0.84       549
     Class 3       0.91      0.84      0.87       591
     Class 4       1.00      1.00      1.00       125
     Class 5       1.00      1.00      1.00       144
     Class 6       1.00      1.00      1.00       145
     Class 7       0.93      0.86      0.89       146
     Class 8       1.00      0.98      0.99       132
     Class 9       0.97      0.98      0.97       153
    Class 10       0.94      0.99      0.96       135
    Class 11       0.74      0.82      0.78       131
    Class 12       0.98      1.00      0.99       154
    Class 13       0.98      0.97      0.97       146
    Class 14       0.99      0.92      0.95       148
    Class 15       0.88      0.84      0.86       146
    Class 16       1.00      1.00      1.00       139
    Class 17       1.00      1.00      1.00       139
    Class 18       0.99      1.00      0.99       136
    Class 19       0.95      0.99      0.97       138

   micro avg       0.94      0.94      0.94      4271
   macro avg       0.95      0.95      0.95      4271
weighted avg       0.94      0.94      0.94      4271
```

Confusion Matrix:

```
[[293   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  1 575   5   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  2  27 469  51   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  1   0  91 499   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0 125   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0 144   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0 145   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0 125   0   0   0  21   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 129   0   0   0   3   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0 150   0   0   0   3   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0 133   0   0   0   2   0   0   0   0   0]
 [  0   0   0   0   0   0   0   9   0   0   0 107   0   0   0  15   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0 154   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   5   0   0   0 141   0   0   0   0   0   0]
 [  1   0   0   0   0   0   0   0   0   0   9   0   0   0 136   0   0   0   2   0]
 [  0   0   0   0   0   0   0   0   0   0   0  16   0   0   0 123   0   0   0   7]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 139   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 139   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 136   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2   0   0   0 136]]
```

This table shows the major misclassifications of 1D CNN:

	Actual				Classified As			
Count	Class	Signal Type	Workload	Defect (inches)	Class	Signal Type	Workload	Defect (inches)
27	Class 2	Baseline	2 HP	N/A	Class 1	Baseline	1 HP	N/A
91	Class 3	Baseline	3 HP	N/A	Class 2	Baseline	2 HP	N/A
51	Class 2	Baseline	2 HP	N/A	Class 3	Baseline	3 HP	N/A
21	Class 7	Faulty	0 HP	0.028	Class 11	Faulty	1 HP	0.028
15	Class 11	Faulty	1 HP	0.028	Class 15	Faulty	1 HP	0.028
9	Class 11	Faulty	1 HP	0.028	Class 7	Faulty	0 HP	0.028
16	Class 15	Faulty	1 HP	0.028	Class 11	Faulty	2 HP	0.028
7	Class 15	Faulty	2 HP	0.028	Class 19	Faulty	3 HP	0.028
237								

There were no misclassifications of baseline that were classified as faulty or faulty that were classified as baseline. All faulty and baseline misclassifications were related to workloads only and were off by 1 HP.

B. XGBoost using Engineered Features Results

The best accuracy score of .90 for the classic approach was XGBoost using a combination of FFT and DWT features. Below is the classification report for XGBoost.

```
Accuracy on training set is : 1.0
Accuracy on test set is : 0.9007258253336455
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       293
           1       0.94      0.97      0.96       581
           2       0.76      0.79      0.77       549
           3       0.85      0.80      0.82       591
           4       1.00      0.97      0.98       125
           5       0.99      0.99      0.99       144
           6       0.99      0.98      0.98       145
           7       0.89      0.89      0.89       146
           8       0.89      0.89      0.89       132
           9       0.97      0.95      0.96       153
          10       0.85      0.90      0.87       135
          11       0.83      0.83      0.83       131
          12       0.90      0.92      0.91       154
          13       0.95      0.95      0.95       146
          14       0.86      0.82      0.84       148
          15       0.89      0.87      0.88       146
          16       0.98      0.99      0.98       139
          17       0.99      1.00      0.99       139
          18       0.93      0.93      0.93       136
          19       0.92      0.93      0.93       138

   micro avg       0.90      0.90      0.90      4271
   macro avg       0.92      0.92      0.92      4271
weighted avg       0.90      0.90      0.90      4271
```

The following is the confusion matrix for XGBoost.

```
[[293   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0 562  19   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0  33 431  85   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0 119 472   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0 121   0   0   0   4   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0 142   0   0   0   1   0   0   0   1   0   0   0   0   0   0]
 [  0   0   0   0   0   0 142   0   0   0   3   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0 130   0   0   0  14   0   0   0   0   0   0   0   2]
 [  0   0   0   0   0   0   0   0 118   0   0   0  13   0   0   0   1   0   0   0]
 [  0   0   0   0   0   1   0   0   0 146   0   0   0   6   0   0   0   0   0   0]
 [  0   0   0   0   0   0   1   0   0   0 121   0   0   0  13   0   0   0   0   0]
 [  0   0   0   0   0   0   0  14   0   0   0 109   0   0   0   8   0   0   0   0]
 [  0   0   0   0   0   0   0   0  10   0   0   0 142   0   0   0   2   0   0   0]
 [  0   0   0   1   0   0   0   0   0   4   0   0   0 139   0   0   0   2   0   0]
 [  0   0   0   0   0   0   0   0   0   0  17   0   0   0 121   0   0   0  10   0]
 [  0   0   0   0   0   0   0   2   0   0   0   8   0   0   0 127   0   0   0   9]
 [  0   0   0   0   0   0   0   0   0   0   0   0   2   0   0   0 137   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 139   0   0]
 [  0   0   0   0   0   0   1   0   0   0   2   0   0   0   7   0   0   0 126   0]
 [  0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   8   0   0   0 129]]
```

The following are the major misclassifications for XGBoost with combined FFT and DWT engineered features.

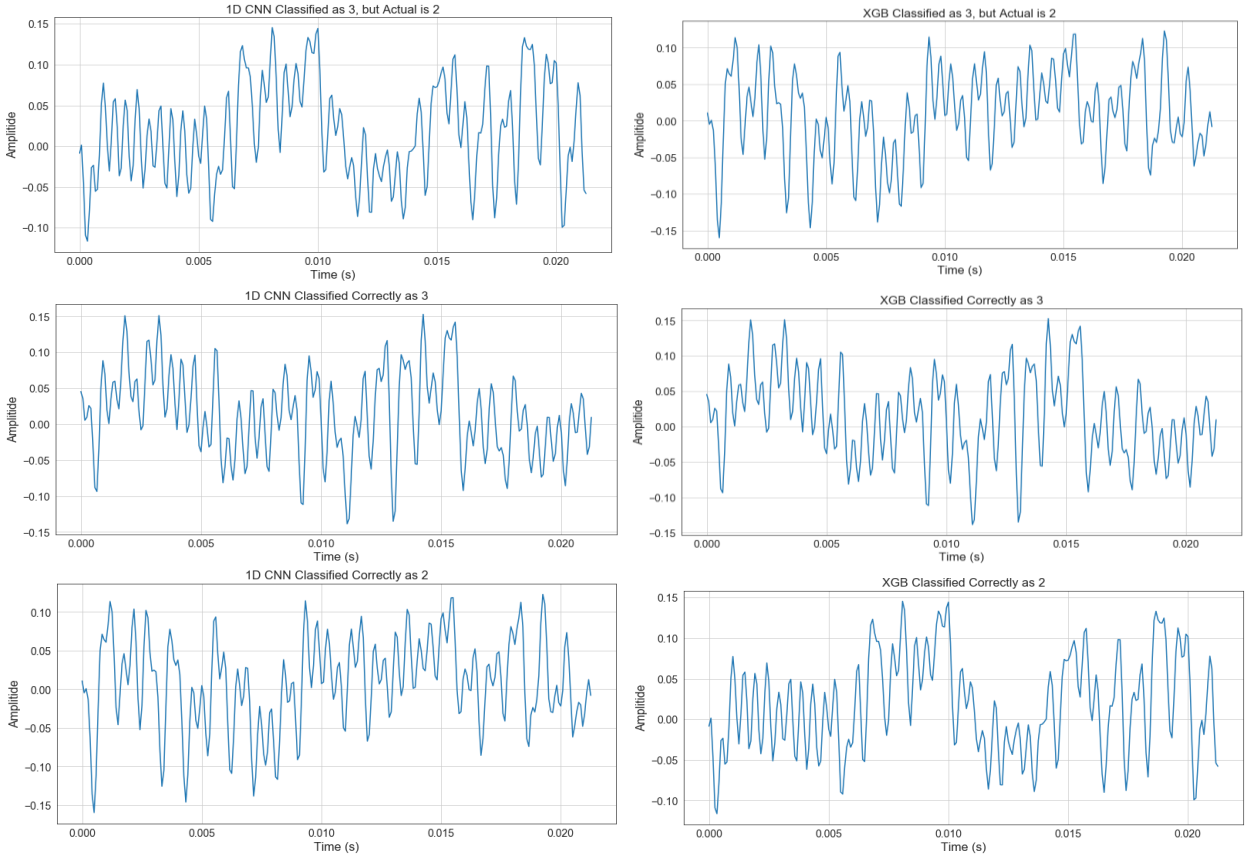
	Actual				Classified As			
Count	Class	Signal Type	Workload	Defect (inches)	Class	Signal Type	Workload	Defect (inches)
33	Class 2	Baseline	2 HP	N/A	Class 1	Baseline	1 HP	N/A
119	Class 3	Baseline	3 HP	N/A	Class 2	Baseline	2 HP	N/A
85	Class 2	Baseline	2 HP	N/A	Class 3	Baseline	3 HP	N/A
19	Class 1	Baseline	1 HP	N/A	Class 2	Baseline	2 HP	N/A
14	Class 11	Faulty	1 HP	0.028	Class 7	Faulty	0 HP	0.028
10	Class 12	Faulty	2 HP	0.007	Class 8	Faulty	1 HP	0.007
17	Class 14	Faulty	2 HP	0.021	Class 10	Faulty	1 HP	0.007
14	Class 7	Faulty	0 HP	0.028	Class 11	Faulty	1 HP	0.028
13	Class 8	Faulty	1 HP	0.007	Class 12	Faulty	2 HP	0.007
13	Class 10	Faulty	1 HP	0.021	Class 14	Faulty	2 HP	0.021
337								

C. Final Assessments

As you can see XGBoost had similar challenges classifying the baseline (normal) sensor data with workloads of HP 1, 2 and 3 applied. Faulty misclassifications had the same defect in inches incorporated into ball bearings but with 1 HP difference between actual and classified results.

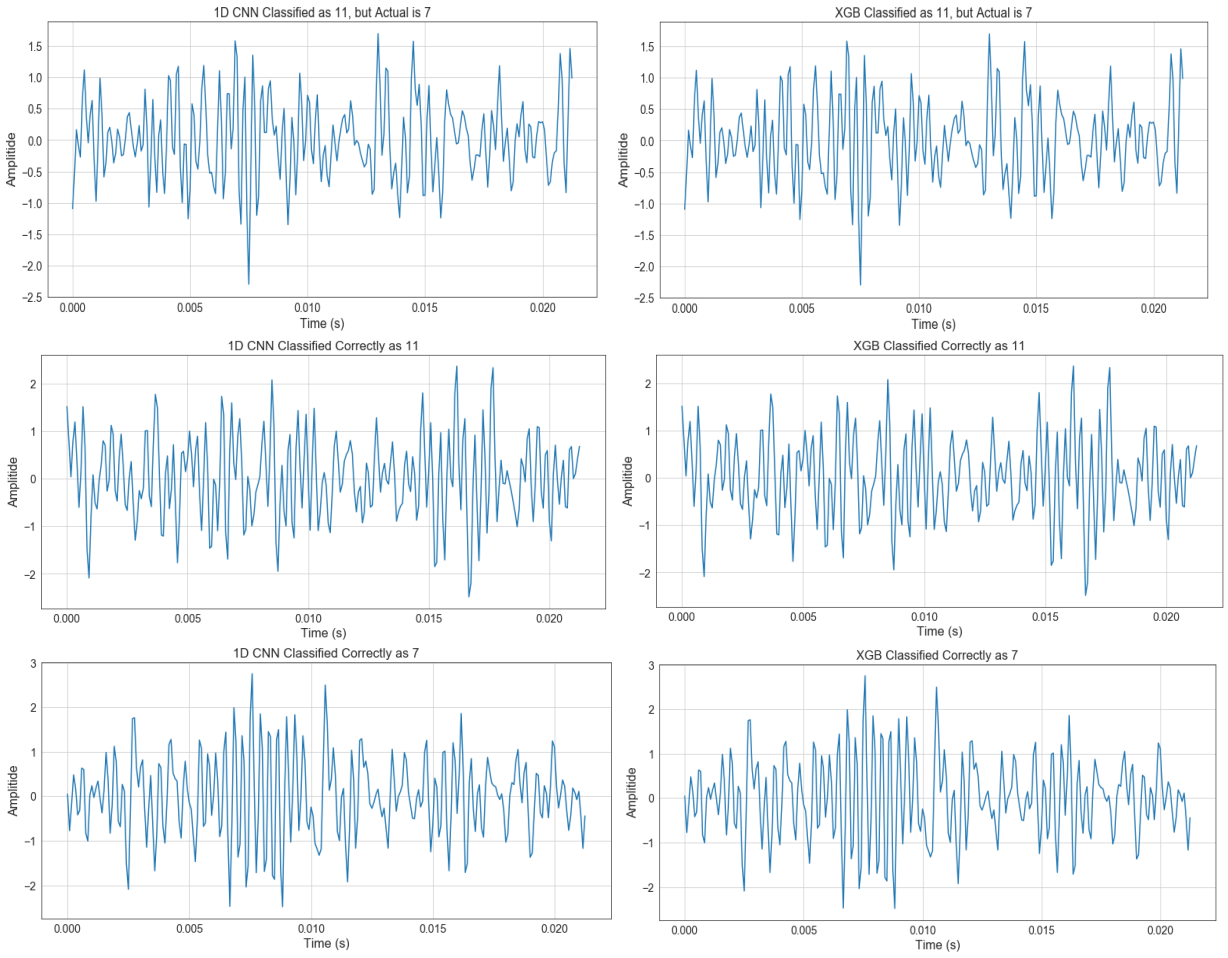
Both approaches had trouble classifying classes 2,3 for baseline and classes 7,8,9,10,11,14 for faulty data. 1D CNN had trouble classifying classes 15 and 19, but XGBoost did not; XGBoost had troubles with classes 1 and 12, and 1D CNN did not.

The following is an example of 1D CNN misclassifying a class 2 as a class 3 as well as XGBoost doing the same. The first column is 1D CNN and second column is XGB results.



Note that the misclassified 1D CNN which classified it as a class 3 but was a class 2 was classified correctly by XGBoost. The last plot (signal) in the second column is the same signal as the first plot (signal) in the first column. Also, note that the misclassified XGBoost which classified it as a class 3 but was a class 2 was classified correctly by 1D CNN. The last plot (signal) in the first column is the same signal as the first plot (signal) in the second column.

The following represents a faulty signal of class 7 which was incorrectly classified as 11 for both 1D CNN and XGBoost.



Note that in this case, both 1D CNN and XGBoost misclassified the same signal and both 1D CNN and XGBoost classified the last two rows of the plot matrix correctly. Note that all three signals (misclassified, actual 11 and classified as 11, actual 7 and classified as 7) are the same signals for 1D CNN and XGBoost.

The conclusion is that both approaches performed well.

IX. Future Considerations

The contemporary approach showed that 1D CNN can be used to accurately predict faulty components in a manufacturing environment using sensor data. Another use case of 1D CNN is for time series forecasting using regression. An interesting discussion at PyData LA 2018 was given by Nathan Janos and Jeff Roach which I highly recommend (https://www.youtube.com/watch?v=nMkqWxMjWzg). Other use cases are natural language processing (NLP), human activity monitoring, patient specific ECG classification, structural health monitoring and anomaly detection in power electronic circuitry. (Kiranyaz, 2019) The knowledge learned in this project can easily be applied to other use cases without the need for significant domain knowledge which is an advantage over the classic approach. However, some domain knowledge is needed. In addition, a benefit 1D CNN's have over other CNN's like 2D CNN's is the amount of data required for training is substantially lower.

In this project, there were 60 FFT engineered features and 84 DWT engineered features for a total of 144 engineered features. Additional features can be added using other feature engineering techniques. For example, "tsfresh" is a python package which automatically creates a large set of time series statistics which can be then used as features for machine learning. It was created by Maximilian Christ of Blue Yonder GmbH and included in the package are seeded datasets which can be used as tutorials. This package can produce 1000's of features and PCA can be used to reduce the set to a manageable level. In addition, wavelet scattering allows for the extraction of engineered features from signal time series and image data. This is fairly new to python and the only information I could find is at the following URL: https://github.com/kymatio/kymatio. Both of these feature extraction methods look to be very interesting and worth trying in the future and likely will improve the results of the classic approach.

X. Supporting Jupyter Notebooks

Signal Analysis for Feature Engineering

Feature Engineering with Bearing Sensor Data

Execution of all Machine Learning Algorithms

XI. Bibliography

Ahamed, N. (2015). *Bearing basics SKF.* Retrieved from https://www.slideshare.net/NaushadAhamed/bearing-basics-skf

CaseWestern. (n.d.). *Case Western Reserve University Bearing Data Center Website.* Case Western Reserve University. Retrieved from https://csegroups.case.edu/bearingdatacenter/pages/welcome-case-western-reserve-university-bearing-data-center-website

Devleker, K. (n.d.). *Understanding Wavlets Parts 1, 2, 3.* MathWorks. Retrieved from https://www.mathworks.com/videos/understanding-wavelets-part-3-an-example-application-of-the-discrete-wavelet-transform-121284.html

Goel, D. (2017). *What is industry 4.0 and how it increases machine efficiency?* Retrieved from https://thingtrax.com/2017/10/05/industry-4-0-increases-machine-efficiency/

IVEDIX. (2017). *The Smart Factory.* IVEDIX. Retrieved from https://ivedix.com/industry-4-0-sensors-analytics-and-the-smart-factory/

Kiranyaz, S. e. (2019). *1-D Convolutional Neural Networks for Signal Processing Applications.* Retrieved from https://ieeexplore.ieee.org/document/8682194

Taspinar, A. (2018). *A guide for using Wavelet Transform in Machine Learning.* Ahmet Taspinar. Retrieved from http://ataspinar.com/2018/12/21/a-guide-for-using-the-wavelet-transform-in-machine-learning/

Taspinar, A. (2018). *Machine Learning with Signal Processing Techniques.* Retrieved from http://ataspinar.com/2018/04/04/machine-learning-with-signal-processing-techniques/

Tower-Clark, C. (2019). *Big Data, AI & IoT Part Two: Driving Industry 4.0 One Step At A Time.* Forbes. Retrieved from https://www.forbes.com/sites/charlestowersclark/2019/02/20/big-data-ai-iot-part-two-driving-industry-4-0-one-step-at-a-time/

Wikipedia. (2019). *Industry 4.0.* Widipedia. Retrieved from https://en.wikipedia.org/wiki/Industry_4.0

Zang, R. e. (2017). *Fault Diagnosis from Raw Sensor Data Using Deep Neural Networks Considering Temporal Coherence.* Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5375835/

XII. Special Thanks

I would like to thanks to Dhiraj Khanna for helping me over the technical hurdles of understanding the Fast Fourier Transform.

I would also like to thank Ahmet Taspinar for the excellent articles and programs describing machine learning related to signal processing and feature engineering using FFT and DWT python functions.