FEATURE EXTRACTION FROM SENSOR DATA (VIBRATION OR MOTION)

Mannar Mannan T
Student of Electronics and communication, Anna university RO Tiruchirappalli.

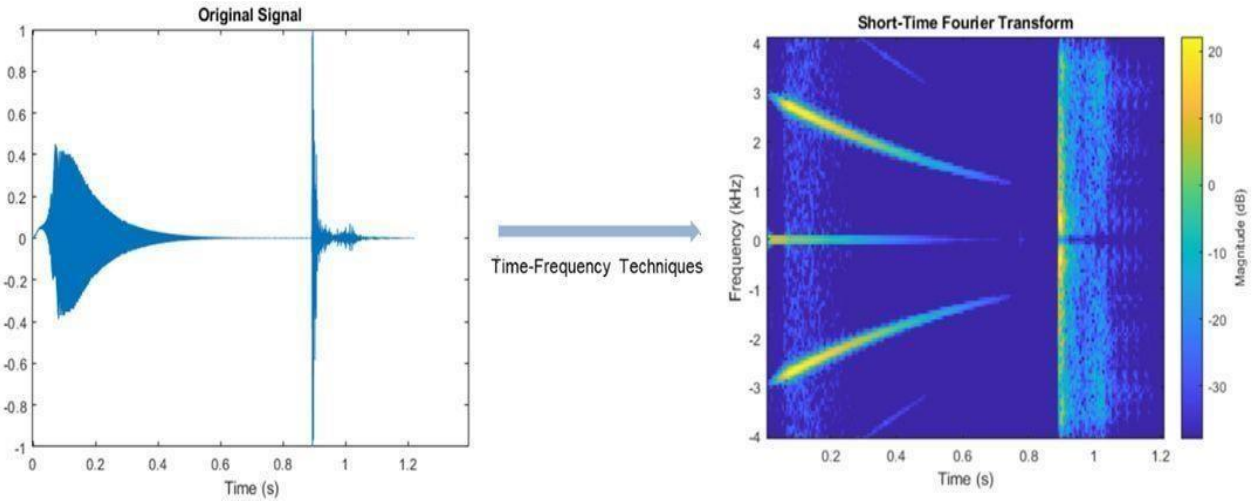
INTRODUCTION

A characteristic of these large data sets is a large number of variables that require a lot of computing resources to process. That's why we use Feature extraction. it refers to the process of transforming and reduce the raw data into a smaller number of features that can be processed while preserving the information in the original data set. It yields better results than applying machine learning directly to the raw data.

Signal features and time-frequency transformations

When analysing signals and sensor data, we can use Signal Processing Toolbox™ and Wavelet Toolbox™ to provide functions that let us measure common distinctive features of a signal in the time, frequency, and time-frequency domains. You can apply pulse and transition metrics, measure signal-to-noise ratio (SNR), estimate spectral entropy and kurtosis, and compute power spectra.

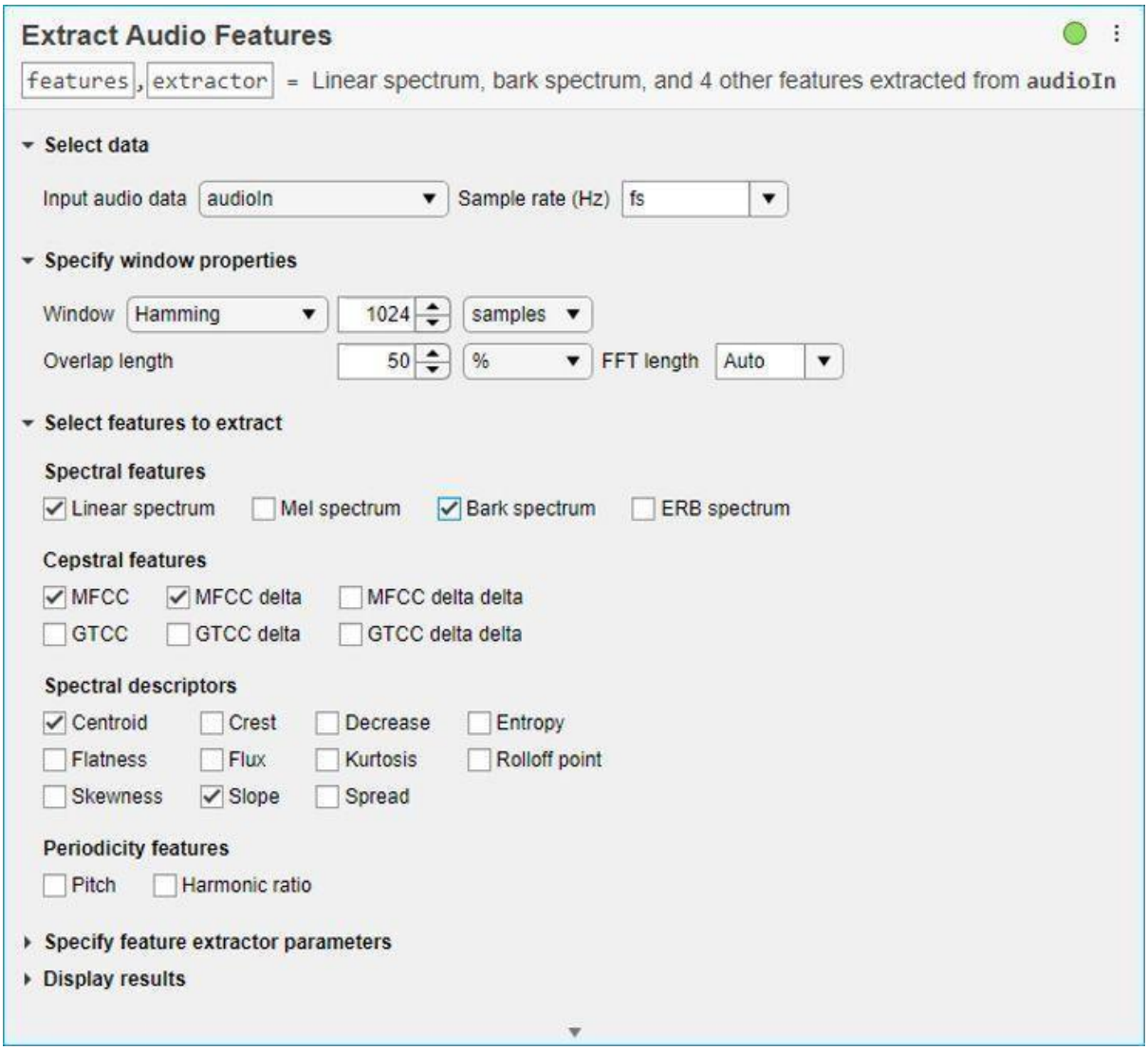
Time-frequency transformations, such as the short-time Fourier transform (STFT) can be used as signal representations for training data in machine learning and deep learning models. For example, convolutional neural networks (CNNs) are commonly used on image data and can successfully learn from the 2D signal representations returned by time frequency transformations.



Spectrogram of a signal using short-time Fourier transform. Spectrogram shows variation of frequency content over time.

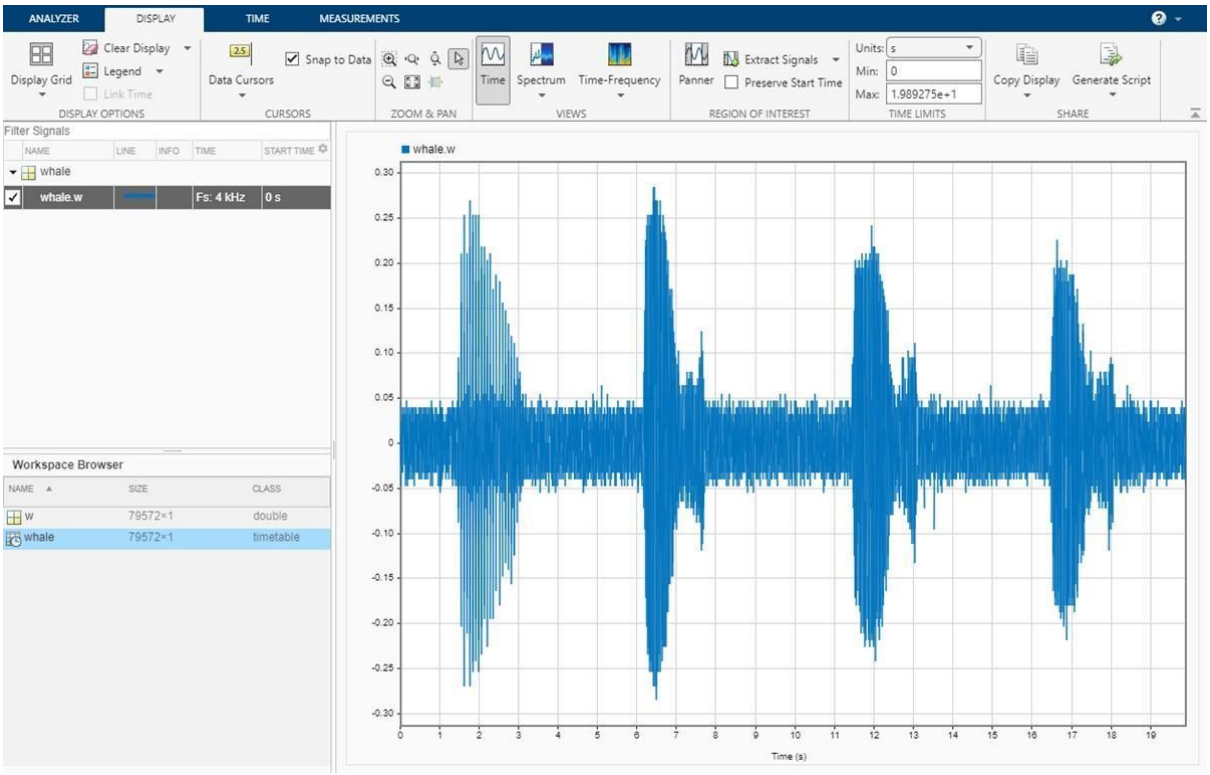
Features for audio applications and predictive maintenance

Audio Toolbox™ provides a collection of time-frequency transformations including Mel spectrograms, octave and gammatone filter banks, and discrete cosine transform (DCT), that are often used for audio, speech, and acoustics. Other popular feature extraction methods for these types of signals include Mel frequency cepstral coefficients (MFCC), gammatone cepstral coefficients (GTCC), pitch, harmonicity, and different types of audio spectral descriptors. The Audio Feature Extractor tool can help select and extract different audio features from the same source signal while reusing any intermediate computations for efficiency.

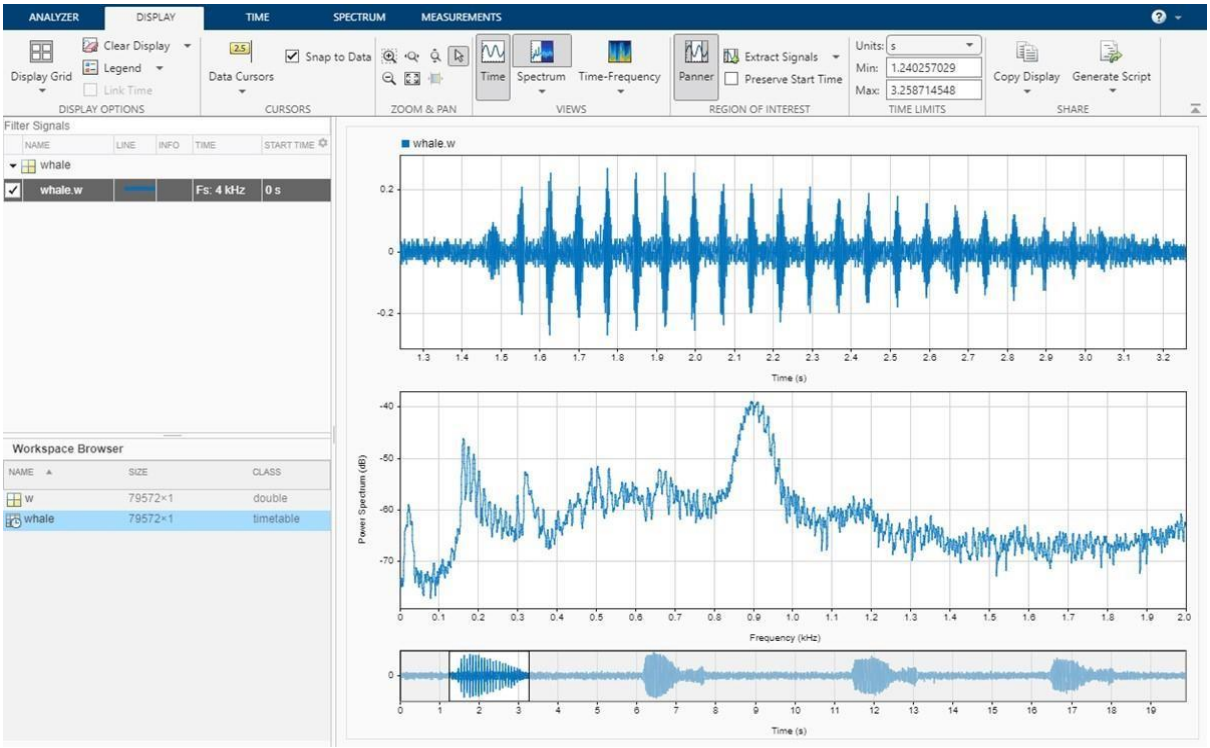


Extract Regions of Interest from Whale Song

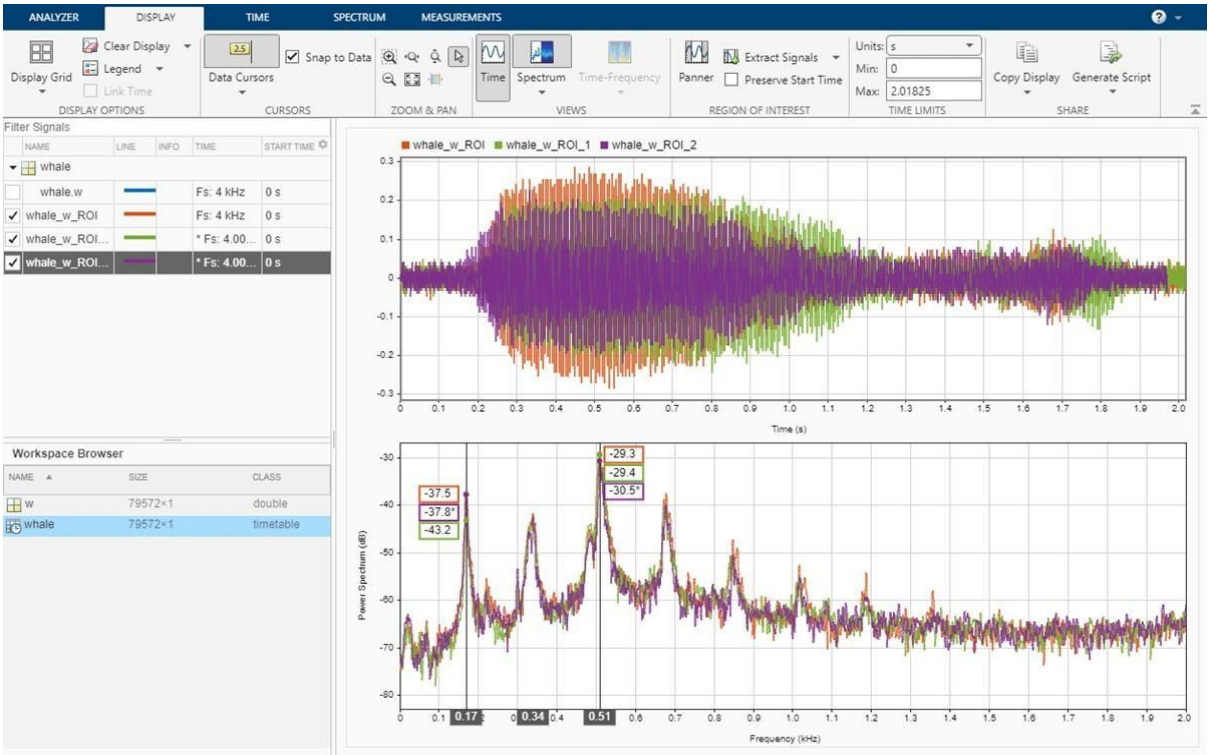
We have audio data from a Pacific blue whale. The file is from the library of animal vocalizations maintained by the Cornell University Bioacoustics Research Program. The raw audio signal is like the below one. its look like complicated doesn't. if we can visually recognize and understand it. It will be easy to machine too.



so, we get 2 seconds of data and do spectrum analysis and in the result the spectrum shows a noticeable peak at around 900 Hz.

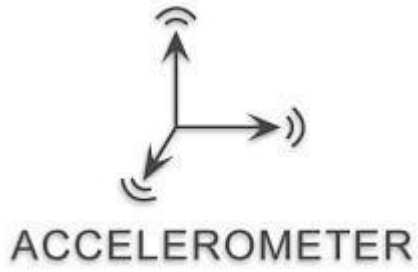


As like above we get 3 samples in size of 2 seconds and we do extraction .so we can compare their spectra of three moans. the three regions of interest you just extracted is displayed below. Their spectra lie approximately on top of each other.



our focus

we specifically going to do feature extraction from sensor (accelerometer) data. That produce 3 data about acceleration in 3 direction like x, y and z. I will show how to process and generate feature from the data



Literature survey

Average

The arithmetic mean is a data set's most commonly used and readily understood measure of central tendency. In statistics, the term average refers to any measurement of central tendency. The arithmetic mean of a set of observed data is equal to the sum of the numerical values of each observation, divided by the total number of observations. Symbolically, for a data set consisting of the values x1,x2,……xn, the arithmetic mean is defined by the formula:

$$\bar{x} = \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right) = \frac{x_1 + x_2 + \cdots + x_n}{n}$$ [3]

Root mean square

The RMS value of a set of values (or a continuous-time waveform) is the square root of the arithmetic mean of the squares of the values, or the square of the function that defines the continuous waveform. In physics, the RMS current value can also be defined as the "value of the direct current that dissipates the same power in a resistor."

In the case of a set of n values $\{x_1, x_2, \ldots, x_n\}$, the RMS is

$$x_{\mathrm{RMS}} = \sqrt{\frac{1}{n}\left(x_1^2 + x_2^2 + \cdots + x_n^2\right)}.$$

The corresponding formula for a continuous function (or waveform) $f(t)$ defined over the interval $T_1 \le t \le T_2$ is

$$f_{\mathrm{RMS}} = \sqrt{\frac{1}{T_2 - T_1}\int_{T_1}^{T_2} [f(t)]^2\,\mathrm{d}t},$$

and the RMS for a function over all time is

$$f_{\mathrm{RMS}} = \lim_{T\to\infty}\sqrt{\frac{1}{2T}\int_{-T}^{T} [f(t)]^2\,\mathrm{d}t}.$$

The RMS over all time of a periodic function is equal to the RMS of one period of the function. The RMS value of a continuous function or signal can be approximated by taking the RMS of a sample consisting of equally spaced observations. Additionally, the RMS value of various waveforms can also be determined without calculus, as shown by Cartwright.[4]

we use root mean square because if we take average of a sine or cosine signal it will be zero because of positive and negative phase of the signal. but in root mean square the values will be squared so the negative became positive so we get the mean value

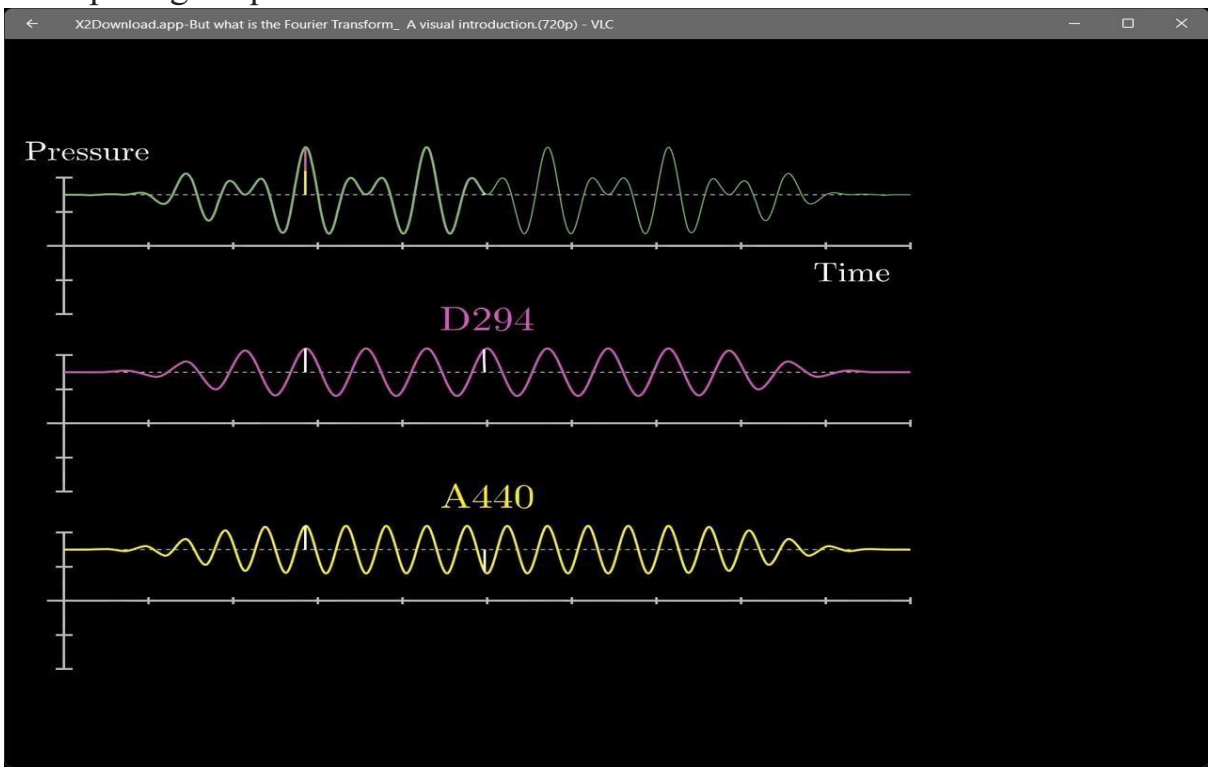
FOURIER TRANSFORM

The Fourier transform is an extension of the Fourier series, which in its most general form introduces the use of complex exponential functions.

Fourier transform

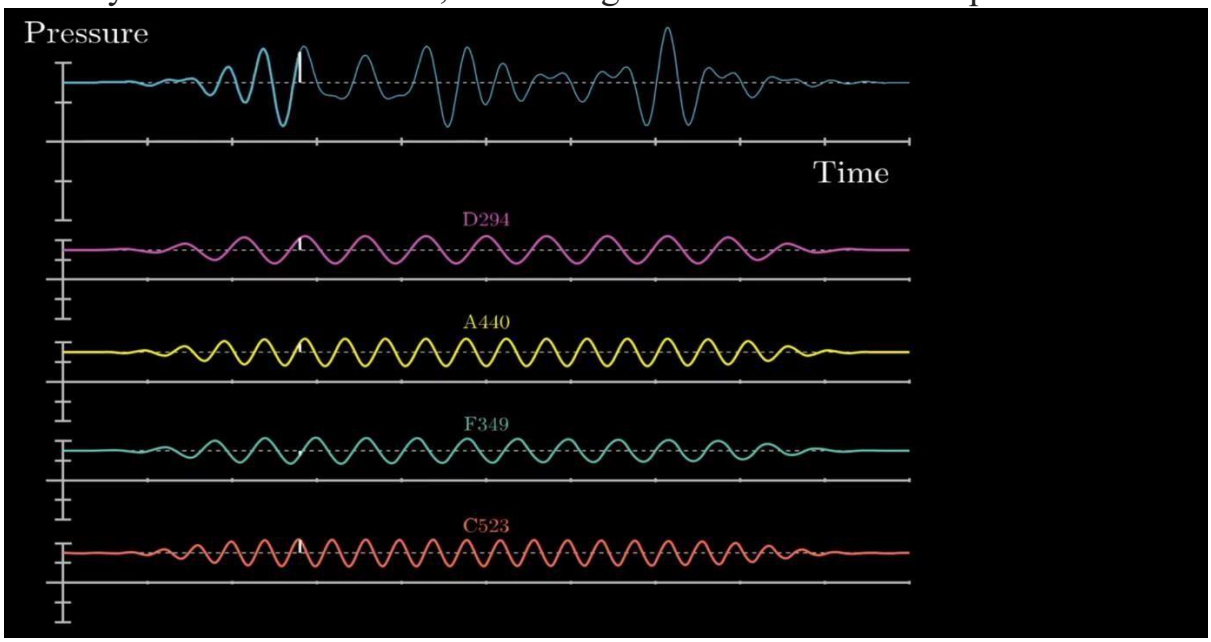
$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)\, e^{-i2\pi\xi x}\, dx. \quad \textbf{(Eq.1)}$$

The Fourier transform is some important mathematical function. I still think that there's something fun and enriching about seeing what all of its components actually look like. The central example, to start, is goanna be the classic one: Decomposing frequencies from sound. Let's dive in.

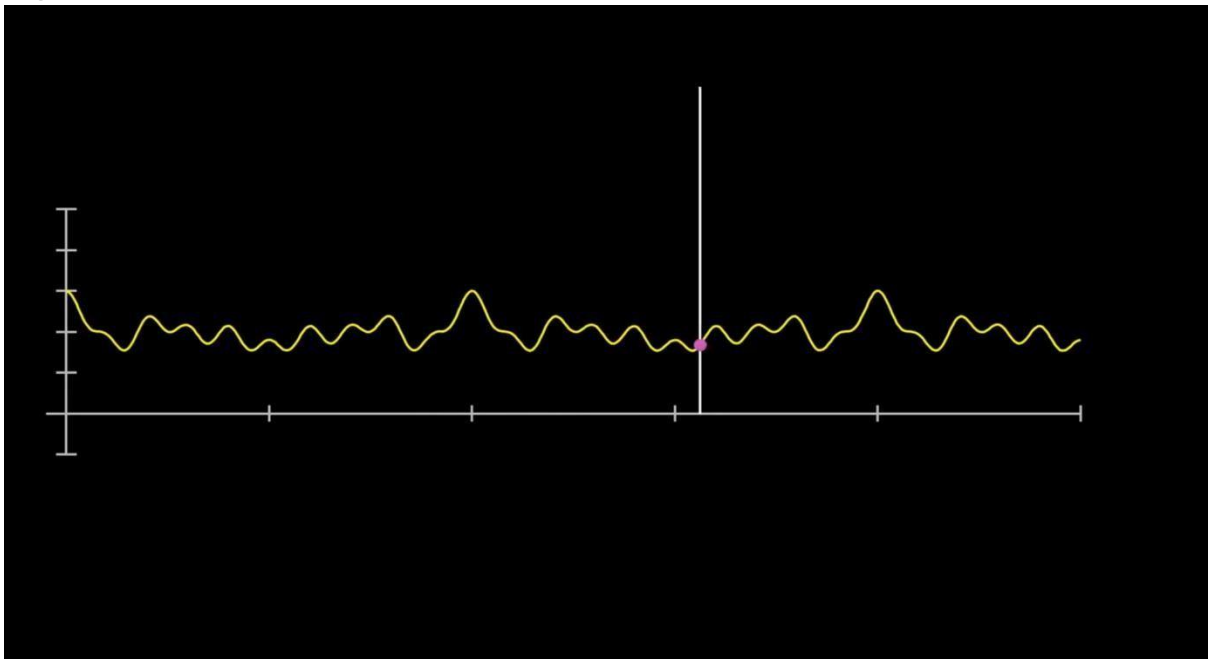


This first sound right here is a pure A 440 beats per second .2nd 440 oscillations each second. A lower-pitched note, like a D, has the same structure, just fewer beats per second. And when both of them are played at once, what do you think the resulting pressure vs. time graph looks like?

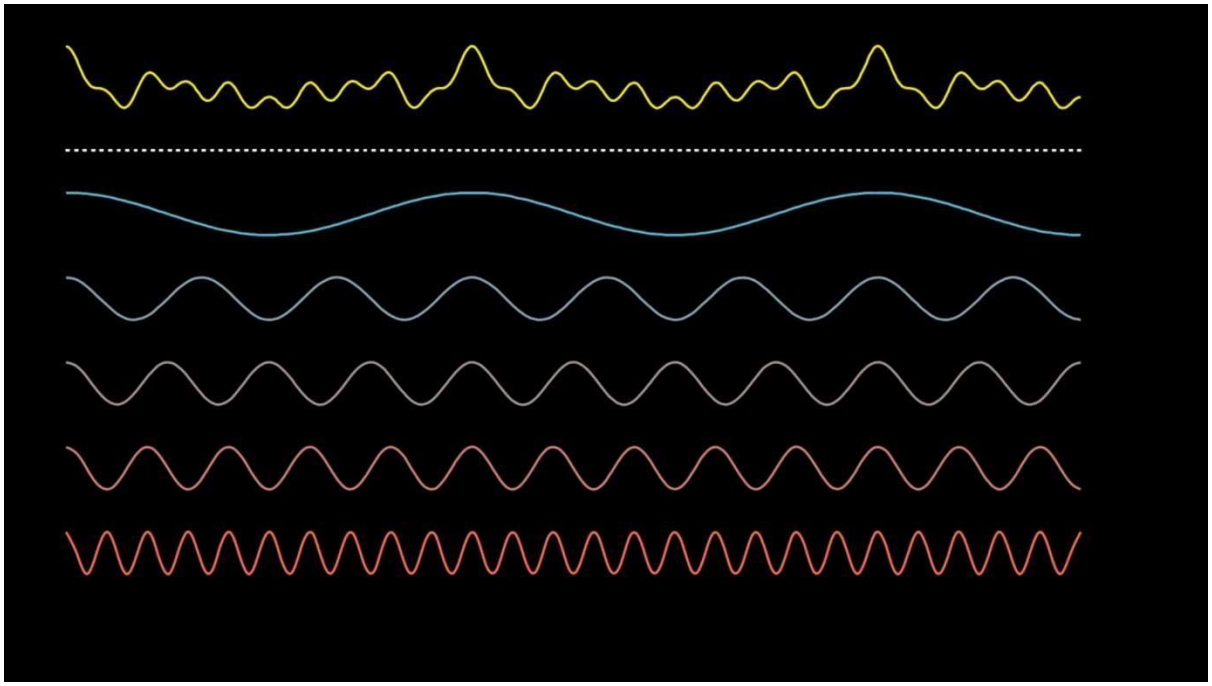
And as you add in other notes, the wave gets more and more complicated.



But right now, all it is, is a combination of four pure frequencies.
a signal like this,

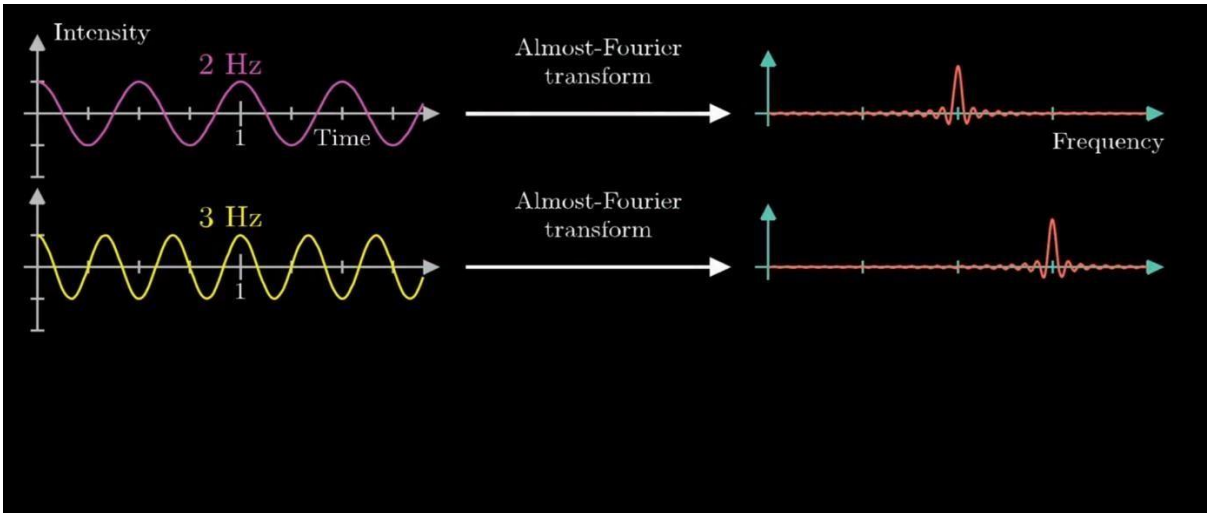


AND DECOMPOSE IT

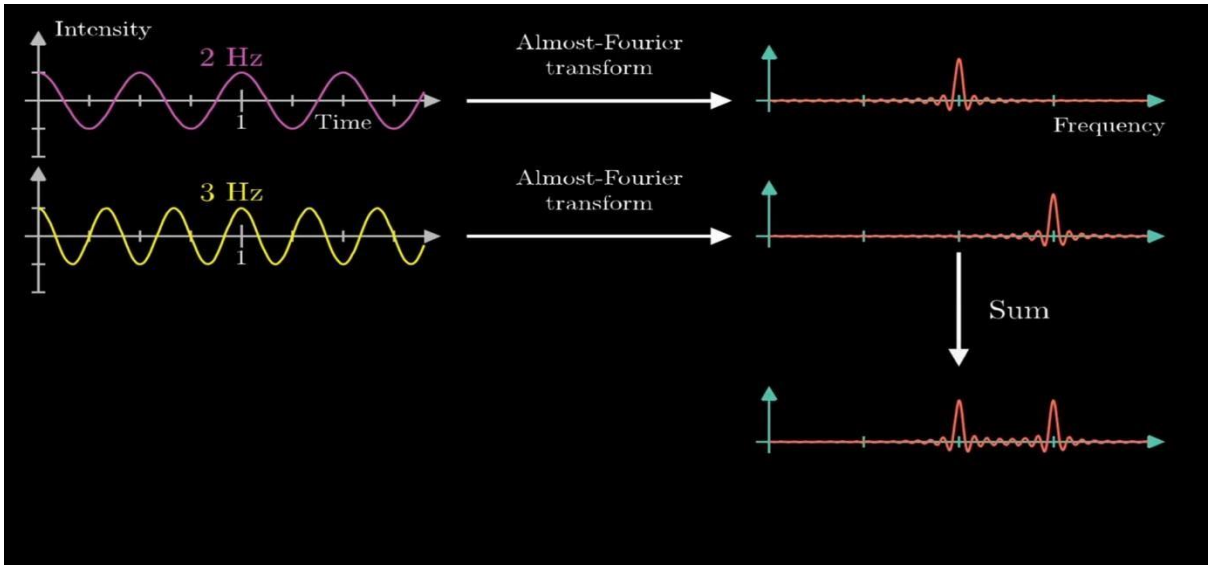


INTO THE PURE FREQUENCIES THAT MAKE IT UP. PRETTY INTERESTING, RIGHT? ADDING UP THOSE SIGNALS REALLY MIXES THEM ALL TOGETHER.SO PULLING THEM BACK APART...FEELS AKIN TO UNMIXING MULTIPLE PAINT COLORS THAT HAVE ALL BEEN STIRRED UP TOGETHER.

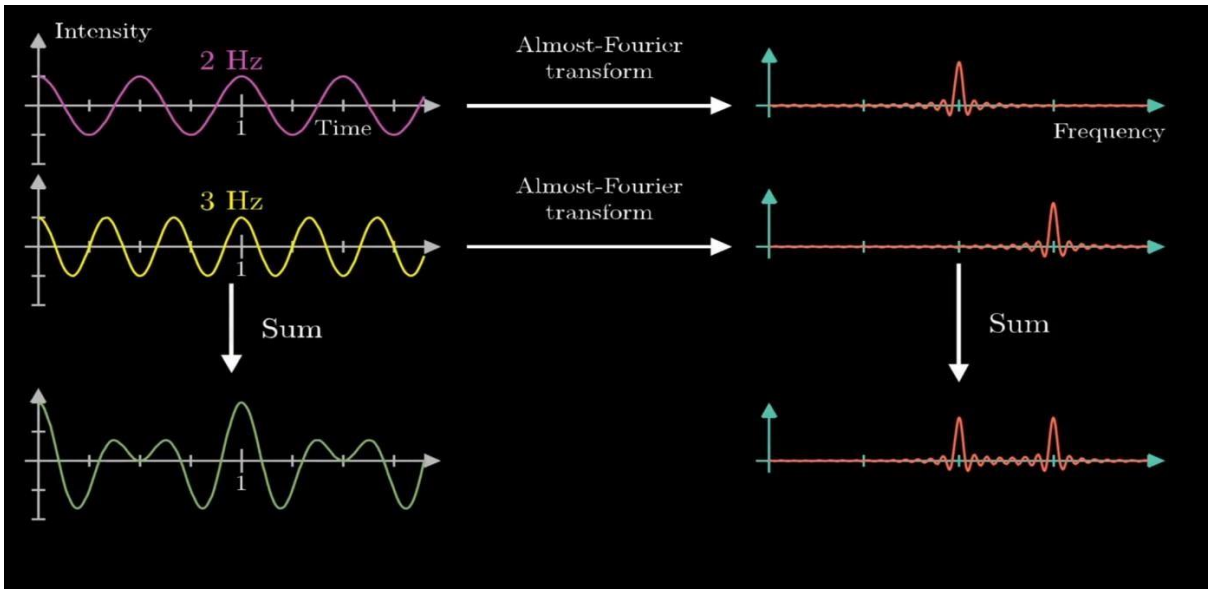
NOW WHAT'S GOING ON HERE WITH THE TWO DIFFERENT SPIKES, IS THAT IF YOU WERE TO TAKE
TWO SIGNALS, AND THEN APPLY THIS ALMOST-FOURIER TRANSFORM TO EACH OF THEM INDIVIDUALLY,



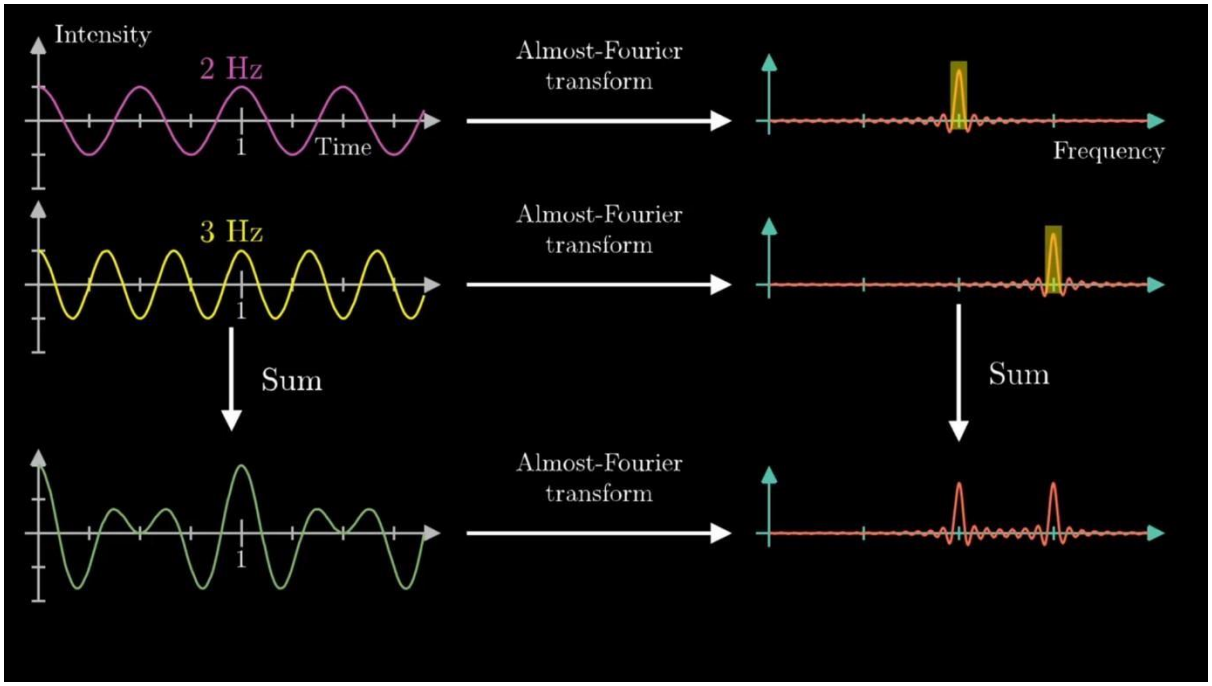
AND THEN ADD UP THE RESULTS,



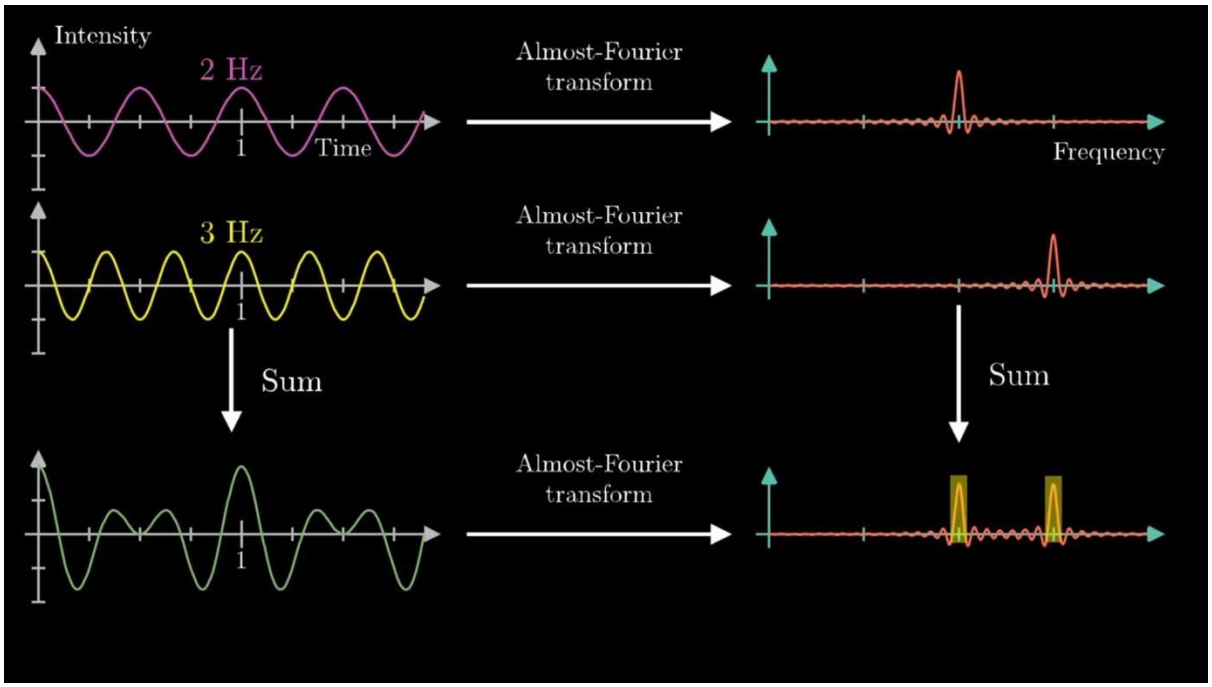
WHAT YOU GET IS THE SAME AS IF YOU FIRST ADDED UP THE SIGNALS, AND THEN APPLIED THIS ALMOSTFOURIER TRANSORM.



THIS PROPERTY MAKES THINGS REALLY USEFUL TO US, BECAUSE THE TRANSFORM OF A PURE FREQUENCY IS CLOSE TO 0 EVERYWHERE EXCEPT FOR A SPIKE AROUND THAT FREQUENCY.



SO, WHEN YOU ADD TOGETHER TWO PURE FREQUENCIES, THE TRANSFORM GRAPH JUST HAS THESE LITTLE PEAKS ABOVE THE FREQUENCIES THAT WENT INTO IT.SO THIS LITTLE MATHEMATICAL MACHINE DOES EXACTLY WHAT WE WANTED.IT PULLS OUT THE ORIGINAL FREQUENCIES FROM THEIR JUMBLED-UP SUMS, LIKE UNMIXING THE MIXED BUCKET OF PAINT.SO THANKS TO
THIS MARVELLOUS ONE WE CAN GET FRQUENCY COMPONETS OF A SIGNAL

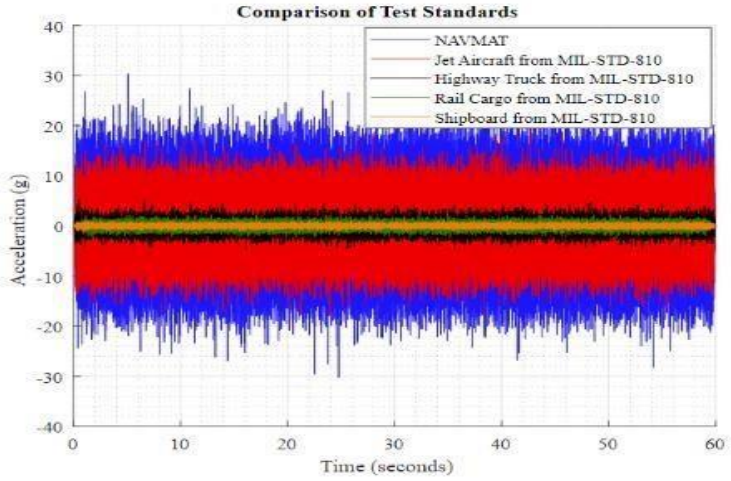


What is a Power Spectral Density (PSD)?

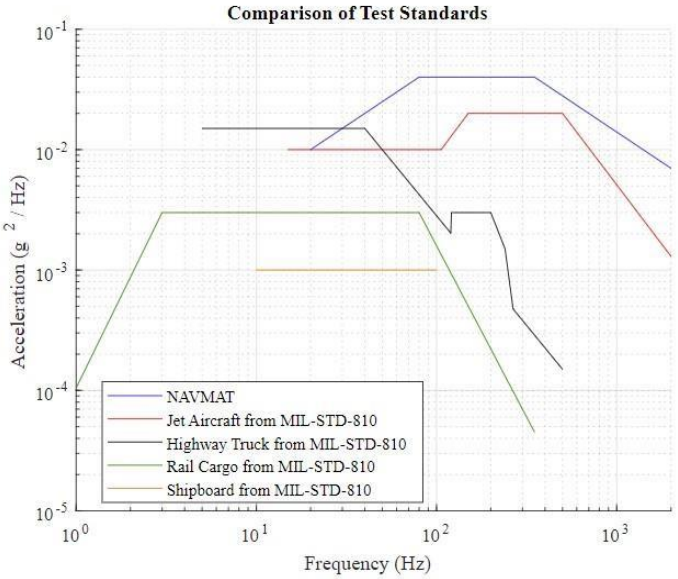
Vibration in the real world is often "random" with many different frequency components. Power spectral densities (PSD or, as they are often called, acceleration spectral densities or ASD for vibration) are used to quantify and compare different vibration environments. Engineers will see PSDs used in test

standards such as MIL-STD-810 and others that provide guidance on how to qualify new products & systems for various operational and transportation environments. I don't want to belabour the math that goes into a PSD (we have our handbook for that!). To illustrate what a PSD is, let's dive into some data! Here are vibration profiles in the time domain over 1 minute for a few different environments:

- NAVMAT (temperature and random vibration screening for Navy contractors)
- Jet Aircraft from MIL-STD-810 (another test standard for military applications)
- Highway Truck from MIL-STD-810
- Rail Cargo from MIL-STD-810
- Shipboard from MIL-STD-810



Looking at this data in the time domain can tell us that the vibration in a jet is likely more severe than rail. But that's only determined from looking at the peak amplitude which isn't always a good indicator. And it doesn't tell us where that energy lies... or what the power *density* is - that's why we have the power spectral density!



here you can see that the PSD plots power (g^2 / *Hz)* in the y axis as a function of frequency (*Hz)* in the x axis. In this format, we can clearly see the distinction between the vibration environments. So, what is this g^2 / *Hz* thing? To answer that question, let's define how a PSD is calculated. The power spectral density function *XPSD(f)* is calculated from the discrete Fourier transform *X(f)* as

$$X_{PSD}(f) = \lim_{\Delta f \to 0} \left[\frac{1}{2} \frac{X(f)X^*(f)}{\Delta f} \right]$$

The one-half factor is needed to convert the amplitude from peak^2/Hz to rms^2/Hz which highlights another benefit of PSDs we'll explore later on. The frequency step is finite in practice and is the inverse of the total measured duration.

$$\Delta f = 1/T$$

This frequency step is the smallest sine wave frequency which can be resolved. A wider Δf gives greater PSD confidence in terms of smoothing the spectral components.

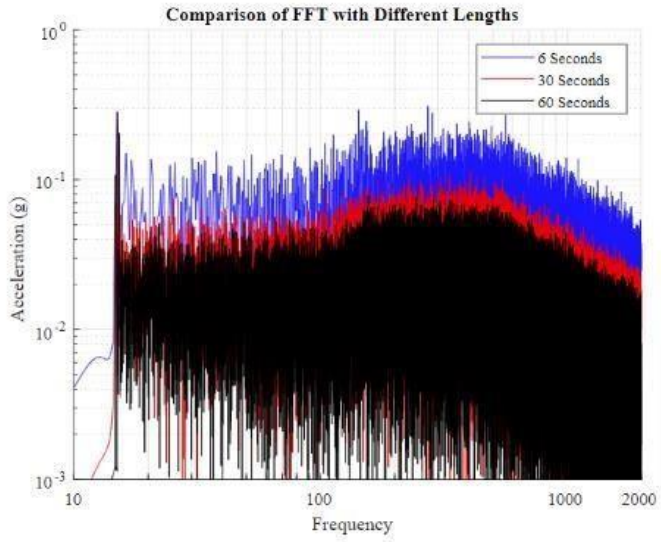
Difference Between PSD and FFT

Although engineers are tempted to use FFTs (fast Fourier transforms) for spectrum analysis, they should really be using (PSDs) power spectral densities. The reason is that PSDs are normalized to the frequency bin width preventing the duration of the data set (and corresponding frequency step) from changing the amplitude of the result. FFTs don't do this!

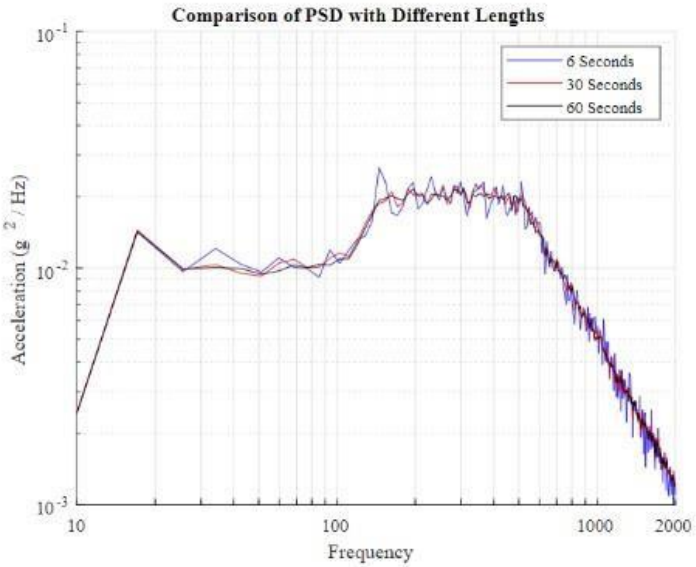
Let's take that same data from the jet standard and calculate and plot FFTs for different time lengths. Check out how the longer the time series is (and the finer the frequency bin width/step is), the lower the amplitude of the FFT becomes!

This means you can't really use FFTs to compare and quantify vibration environments

.



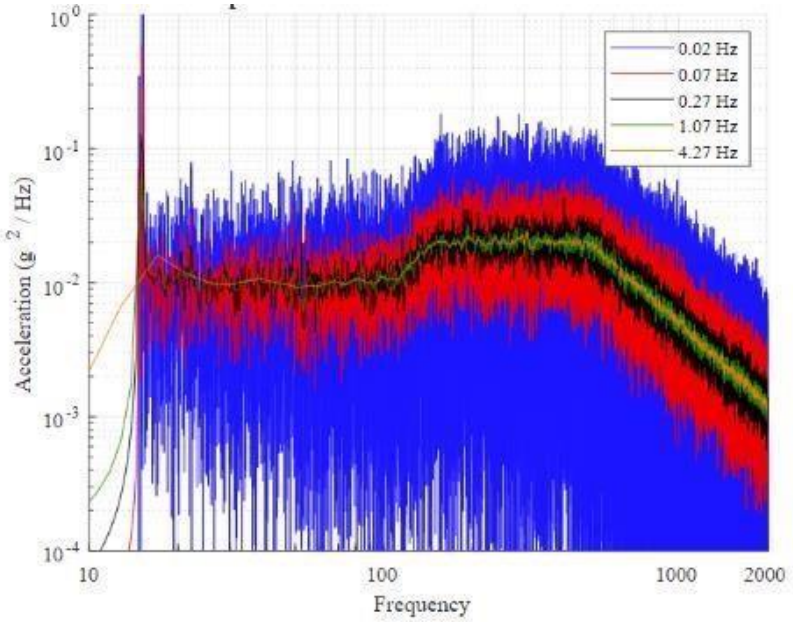
Now let's do that same calculation for the PSD. In this plot I've kept the frequency bin width the same at 8 Hz. These different time lengths do nothing to the amplitude of the resulting PSD because they are normalized to the bin width.



Benefits and Features of PSDs

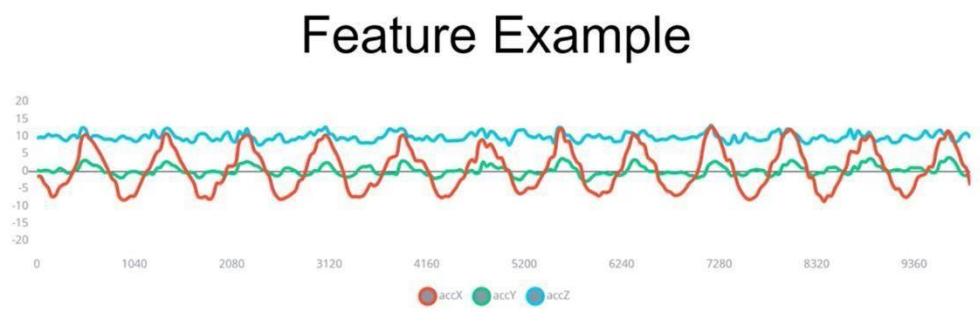
Adjustable Width to Smooth Bin

You probably noticed in the last section how amazingly clean those PSD were! This is another major benefit of PSDs because they are dependent on that frequency bin width. You can manipulate this to smooth a PSD. The VibrationData Toolbox has a great feature that I highlight in our article on generating a PSD test standard. Here I highlight how changing that bin width from a very fine 0.02 Hz to 4 Hz will drastically smooth the PSD while keeping the overall energy in each frequency accurate and consistent across the whole range.



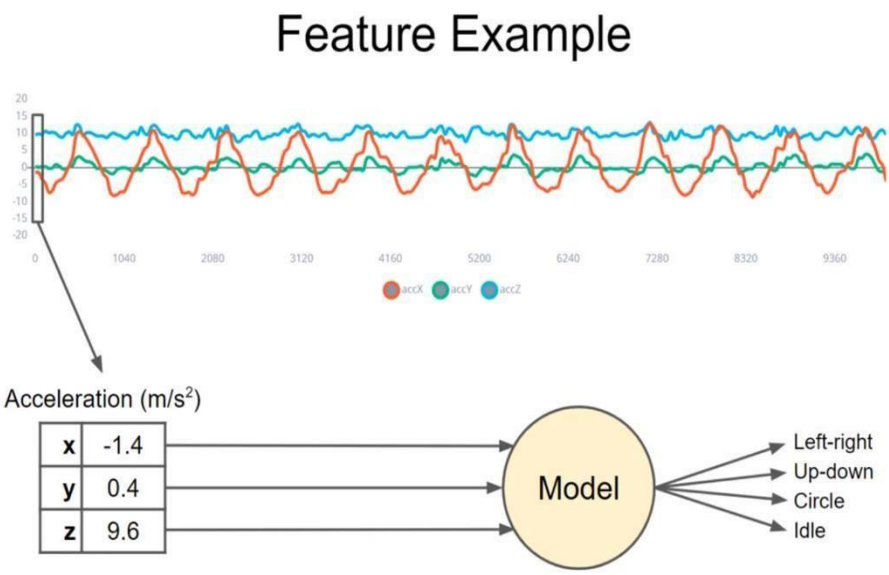
METHODOLOGY

One of the most important things we can do in machine learning, is figure out which features to extract from our data to send to our model for training and, ultimately, inference. Many modern deep learning techniques do allow us to send raw data right to the model, and have the model figure out which features it cares about. But that often requires a lot more processing power than we have available in some of our embedded systems. At this point, you may be wondering, what is a feature? For machine learning, a feature is an individual measurable property or characteristic of a phenomenon being observed. Let's start with a simple example.

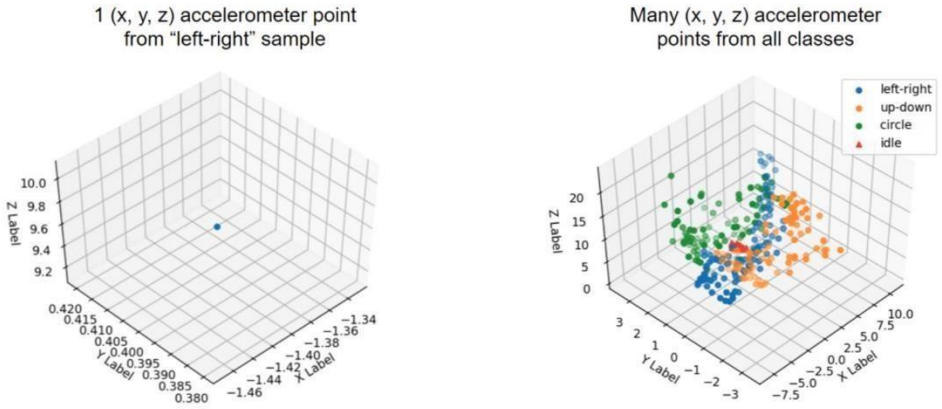


Here is a plot of one of our ten-second motion examples we gathered earlier. By looking at it, can you tell which motion it is? Look at the legend to see what's moving the x, y, or z axis. This is the left to right motion as the x-axis plot tells us it was moving back and forth. Now, let's start by looking at a very simple example of a feature set. We'll take a single point in time and look at the x, y, and z acceleration data at that point.

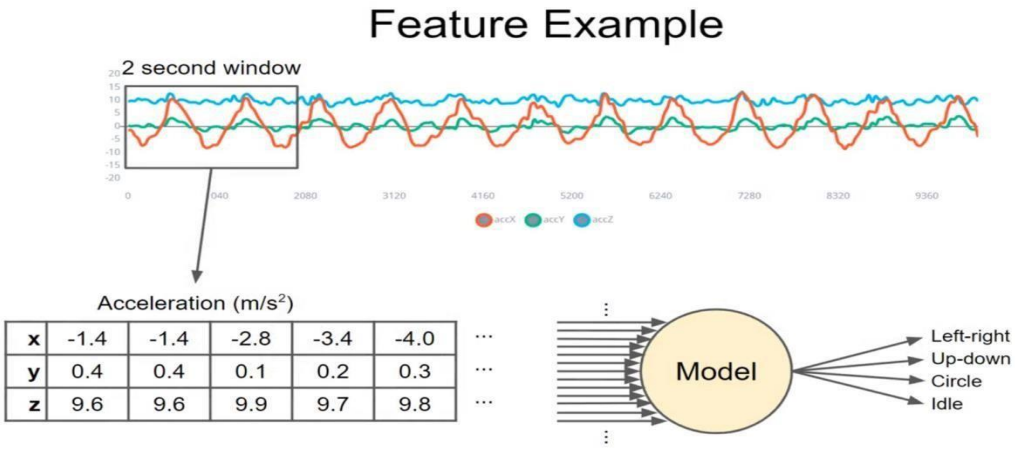
We could, potentially, use this as a simple feature set. Each value in the feature set makes up a dimension in the input to our machine learning model. So, we'd say that this feature set has three dimensions.



Our machine learning model would then, always, expect three inputs. We would use these features and associated labels to train the model. When it's done training, the model would, once again, look for three features as inputs, and use those three features to try to predict the class that they belong to. Do you think that these features make for a good feature set for our model to predict these particular motions? Why or why not? These are actually pretty poor features, because they do not take a time sequence into account. It's just a snapshot in time of what the accelerometer is seeing. The input to the model is just this snapshot. There's no information in these features about how the acceleration changes over time. The nice thing about three dimensions is that we can plot that point pretty easily and visualize what's going on. Now let's plot similar points taken from different samples from each of our four classes.

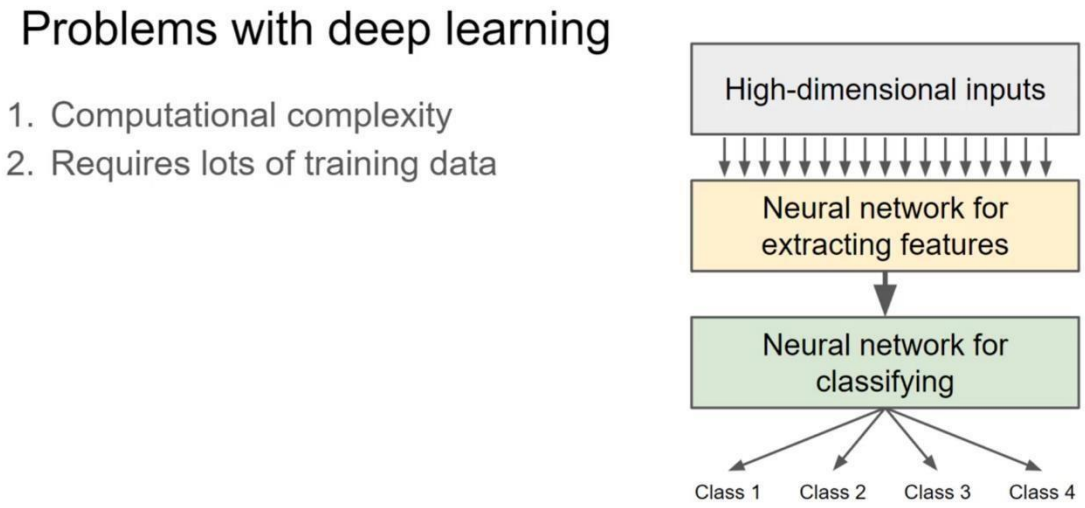


I've made each of the classes a different colour. It's a little messy, but you can see that there are distinct shapes. Most of the time, with classification problems in machine learning, if you can visually separate the clusters of samples, you can train a machine to do the same. However, if you look at the centre of the group, you can see that all of the samples merged together in one clump. This is where the x, y, and z acceleration in each of the four motions, looks about the same. So, there are some snapshots in time where all of the classes look about the same, and the machine learning model will have a hard time discerning the difference among them. One way to fix this is to use the points in a longer window of time as inputs. Let's make our window long enough to capture one or two cycles of the movement, which would be about two seconds in each axis.



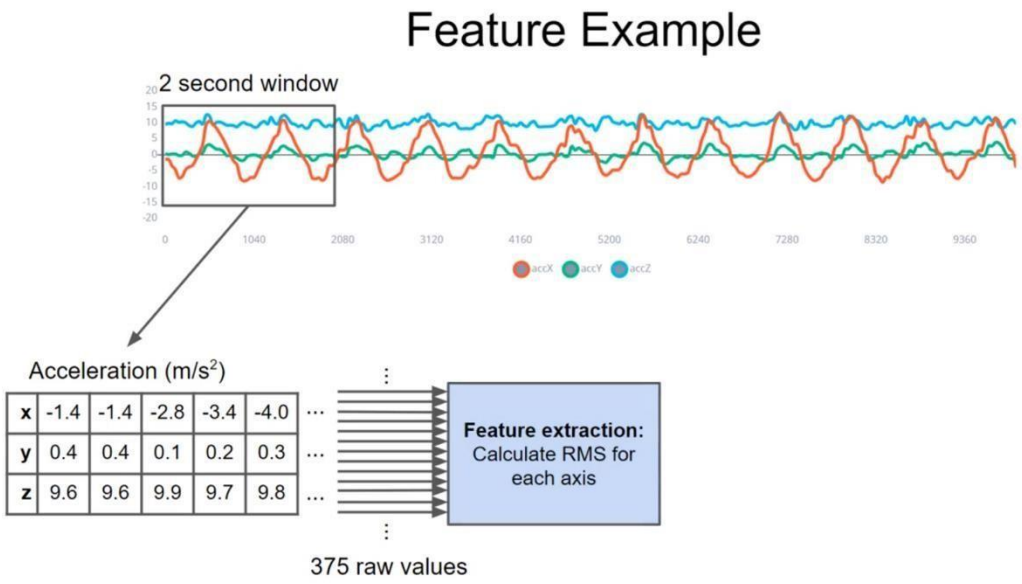
With a sampling rate of 62.5 hertz, that gives us 125 samples for each axis. So, in total, that's 375 data points being fed into the machine learning model. Unfortunately, this means our input matrix to the model is now 375 dimensions, which is much harder for humans to visualize. Many machine learning

models, like deep neural networks, are capable of extracting the necessary features from large inputs like this, automatically. They can take large input arrays and learn what to look for to make good decisions. However, there are two main problems with this approach. Here is an example of a deep learning model.



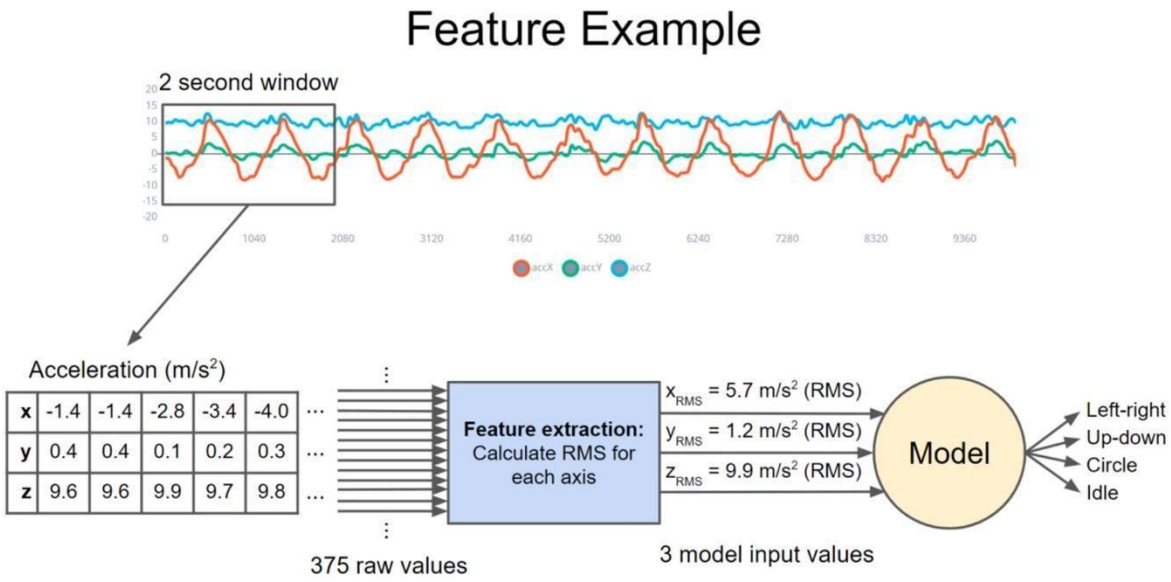
These networks can take images which consists of hundreds or thousands of pixels corresponding to hundreds or thousands of dimensions, figure out what features to look for in those images, and then classify the image based on those features. The problem is that, as the machine learning model grows in size and complexity, so, do the computational requirements. It takes a lot of processing power to automatically learn what features are useful for the classifier, and then use that feature extractor in real time. With a 1,000-dimension input, you're going to need to store, at least, that amount of data in memory, and be able to perform mathematical operations on each of those values.
Feature extraction sections in such a model can often be several layers deep, multiplying the computational complexity required to perform inference. Additionally, as we make machine learning models larger, then generally require a lot more data to train. Often, we don't have an endless supply of training data and an endless amount of time for training. While such deep learning models may be great for huge server farms, doing things like natural language processing, we don't have that luxury in embedded systems. Even though we can, and will, use deep neural networks in this class, you will benefit greatly by choosing appropriate features from the raw data. You ultimately want to keep your machine learning model as small and fast as possible. As most

machine learning algorithms require a lot of memory and processing power, and those are in limited supply in most embedded systems. One way we can do this, is by choosing the features for the model, manually. Rather than picking and choosing some of the raw samples, we can combine the samples in a variety of ways to generate unique features that help describe what's going on in the system. For example, maybe we calculate the root mean square for all 125 samples in each axis.

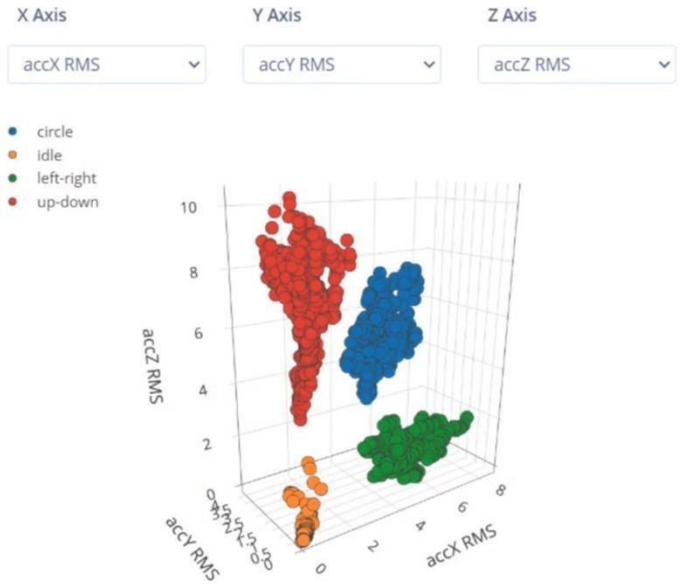


This is a straightforward calculation that gives us a single number per axis, and gives us something like an average or mean, for all these numbers in each axis. We're back to having only three dimensions as the input for our model, and they take into account two seconds of data. One thing you might want to do is filter out the mean of each set of data before calculating the root mean square, so that gravity is taken out of the equation

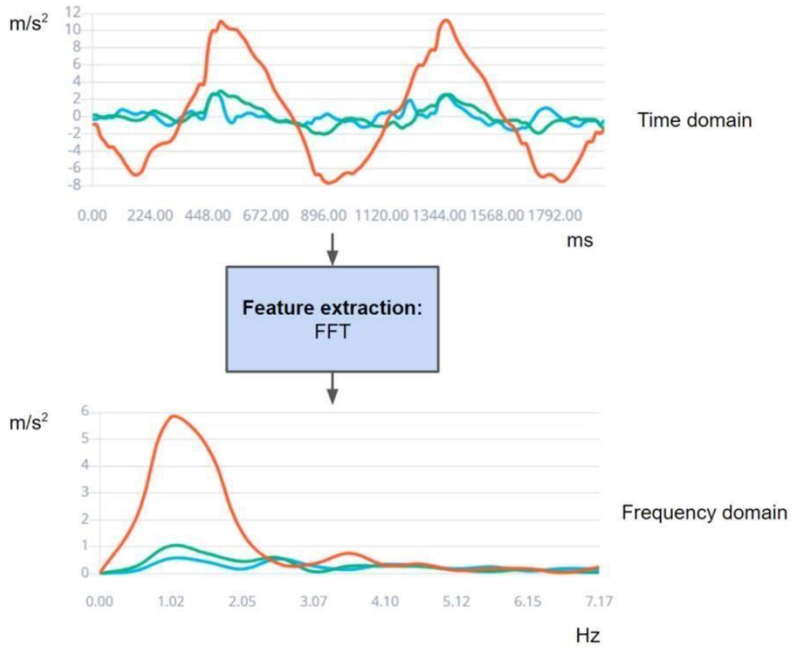
.



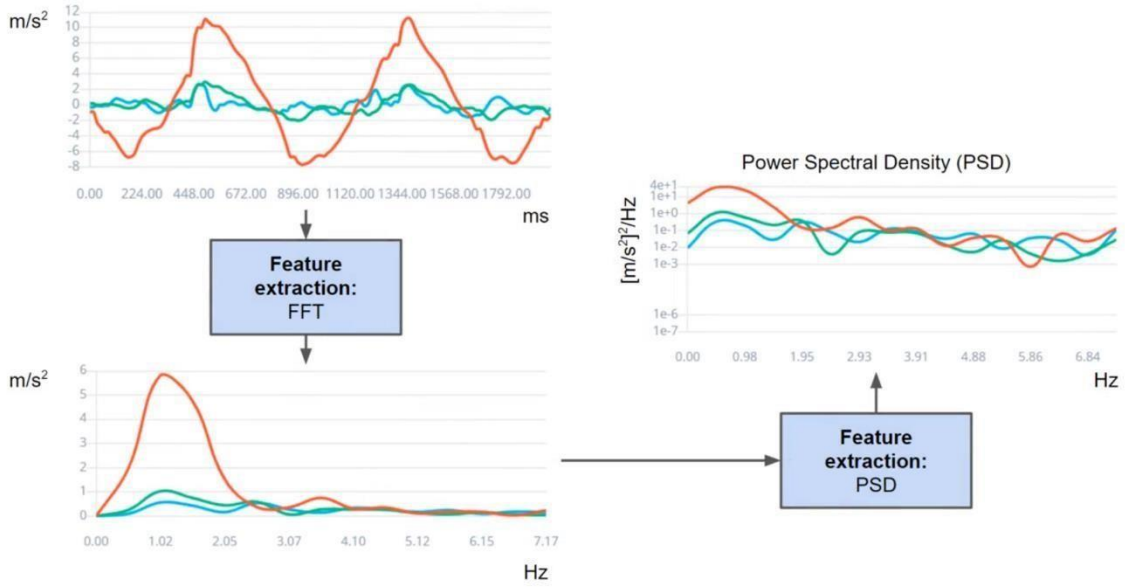
We could probably create a classifier model that works very well on just RMS data, as there is a lot of separation between these feature groups.



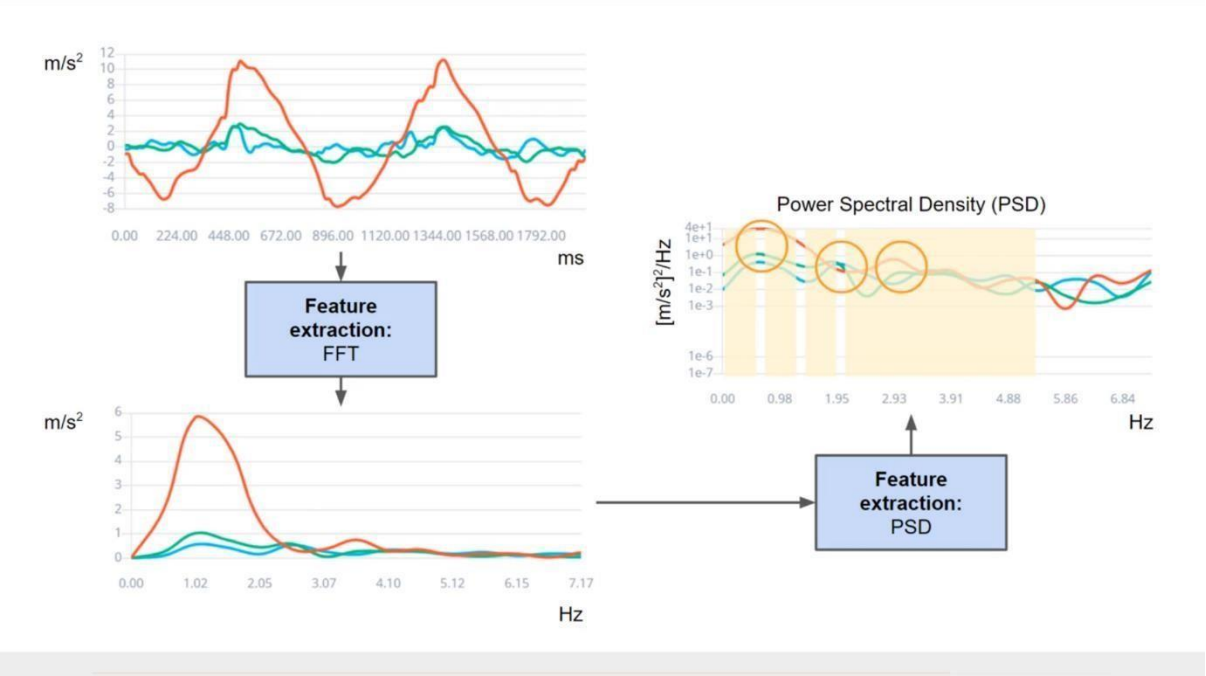
But we can take this a step further. One common technique when looking at vibration or motion data, is to take the Fourier transform of that data to get information about it in the frequency domain. We won't go into detail about how to calculate the Fourier transform, but know that it's a way to break apart a signal into its various frequency components. The Fast Fourier Transform or FFT is optimized for discrete sampled data like what we have here.



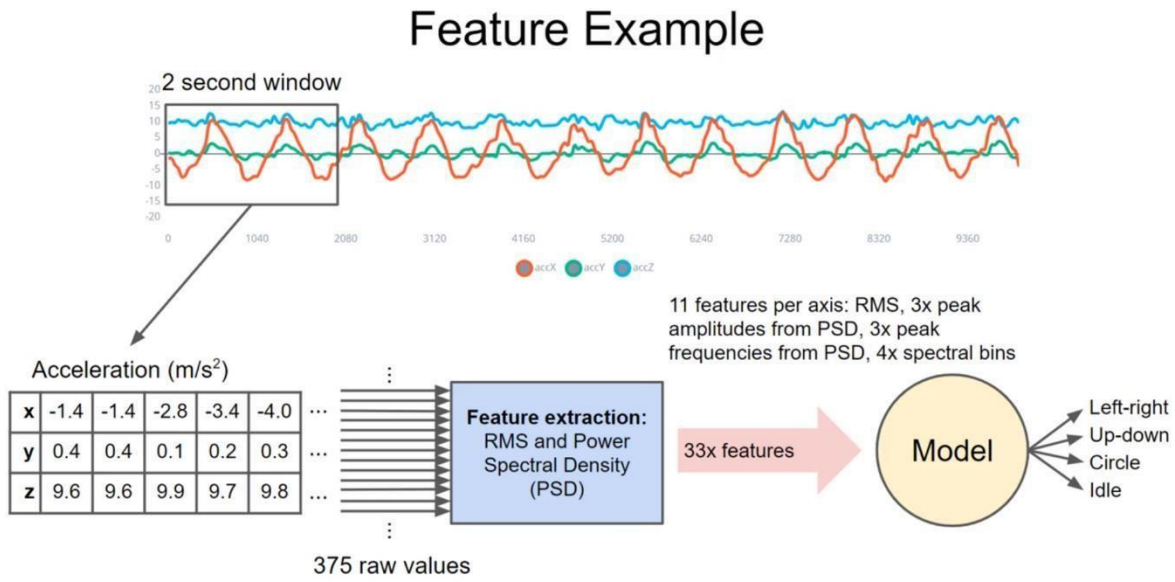
This is an example of the left to right motion. If you look closely at the repeating left to right plot on the top graph, you can see that it takes about one second to complete one cycle. That means it has a frequency of one hertz. Because this is a prominent feature, it will stand out in the frequency domain. Take a look at the bottom graph and you'll see a peak at one hertz, noting that there is a large one hertz component. Similarly, if you were to move the phone back and forth faster at, say, three hertz, this peak would then be at three hertz. Edge Impulse takes this even another step further and computes the Power Spectral Density or PSD from the FFT.



The FFT can be somewhat misleading sometimes, as the amplitude of the FFT in each bin can vary based on the frequency width of that bin. The PSD helps by normalizing the amplitude to the width of each bin. Rather than feed all of the values in the PSD to the model, Edge Impulse extracts a few important characteristics from it.



The first, is identifying the amplitude and frequency location of the highest peaks of each axis. The second, is they sum all of the values between certain ranges in the PSD. These two actions help describe the shape of the PSD without needing to use all the values in it, which would require a higher dimension input to the model, and we want to avoid that. Just know that the power spectral density is another good set of features, especially, when it comes to motion and vibration data. Edge Impulse is taking 375 raw samples from our 2 second window and computing a number of features.



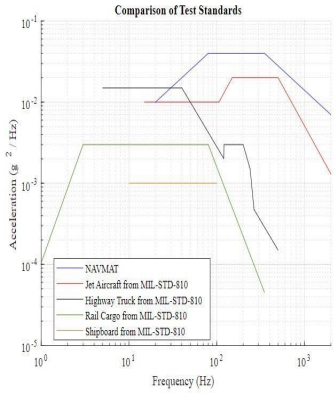
From each axis, we'll send the RMS value, three peak amplitudes from the PSD, three peak frequency locations from the PSD, and four summed bins from the PSD to the model. So rather than have a 375-dimensional input to our model, we have a 33-dimensional input, which is much better.

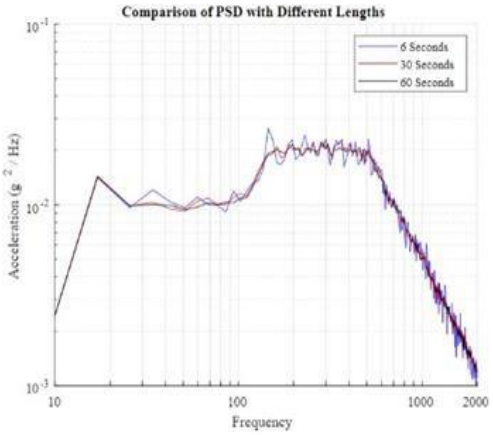
Result

By doing feature extraction we particularly extract the parametrical data from the original data. because unlike like see from single side we choose the best view parameter and specifically extract that and look from that way so we can easily understand and also machine can also easily understand too and get the best result

Parameter of interest

As you can see from the above details that power spectral density is best for vibration data analysis .and look upon our data is acceleration and it is just like vibration so PSD is best. The reason is that PSDs are normalized to the frequency bin width preventing the duration of the data set
(And corresponding frequency step) from changing the amplitude of the result





CONCLUSION

So, this is the end of this, we discussed about the feature extraction from the acceleration data. And the steps to do that and benefits and how its benefit and what are the difference like wise. And that all now