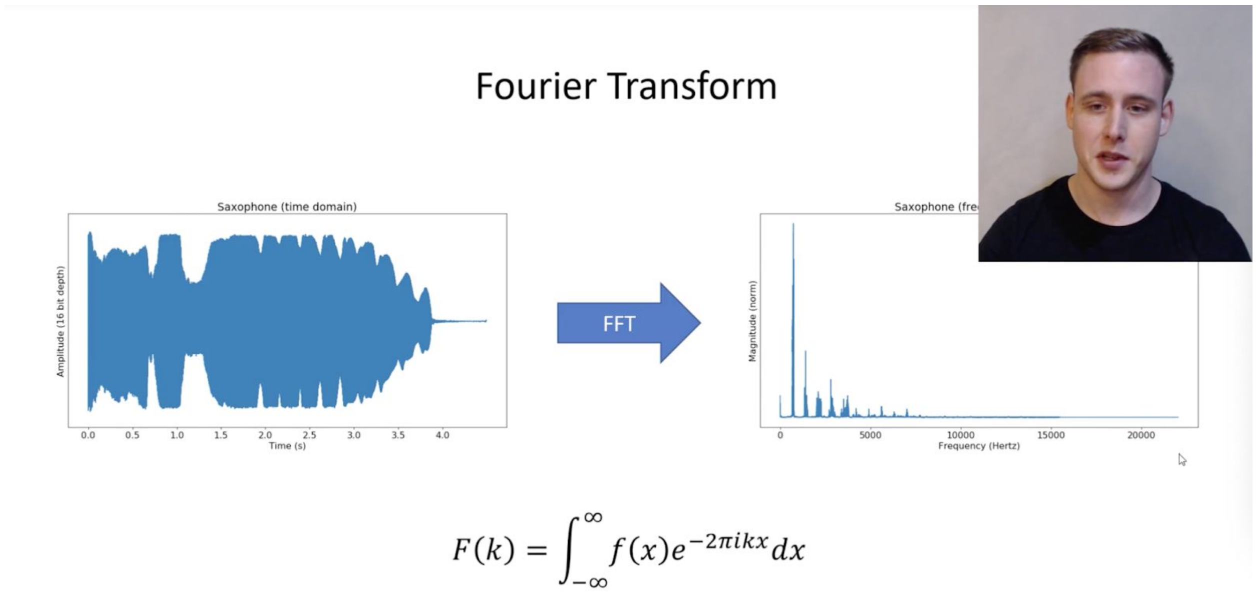
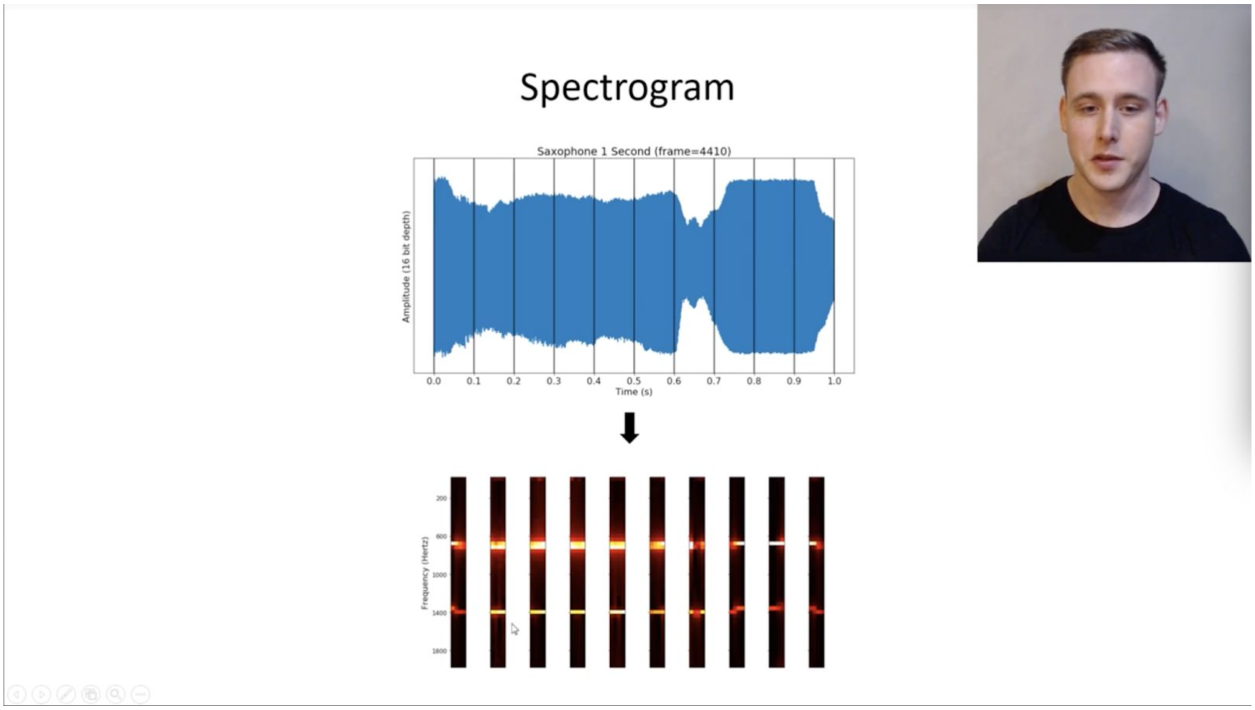
Audio Classification
Acknowledgements: Seth Adams

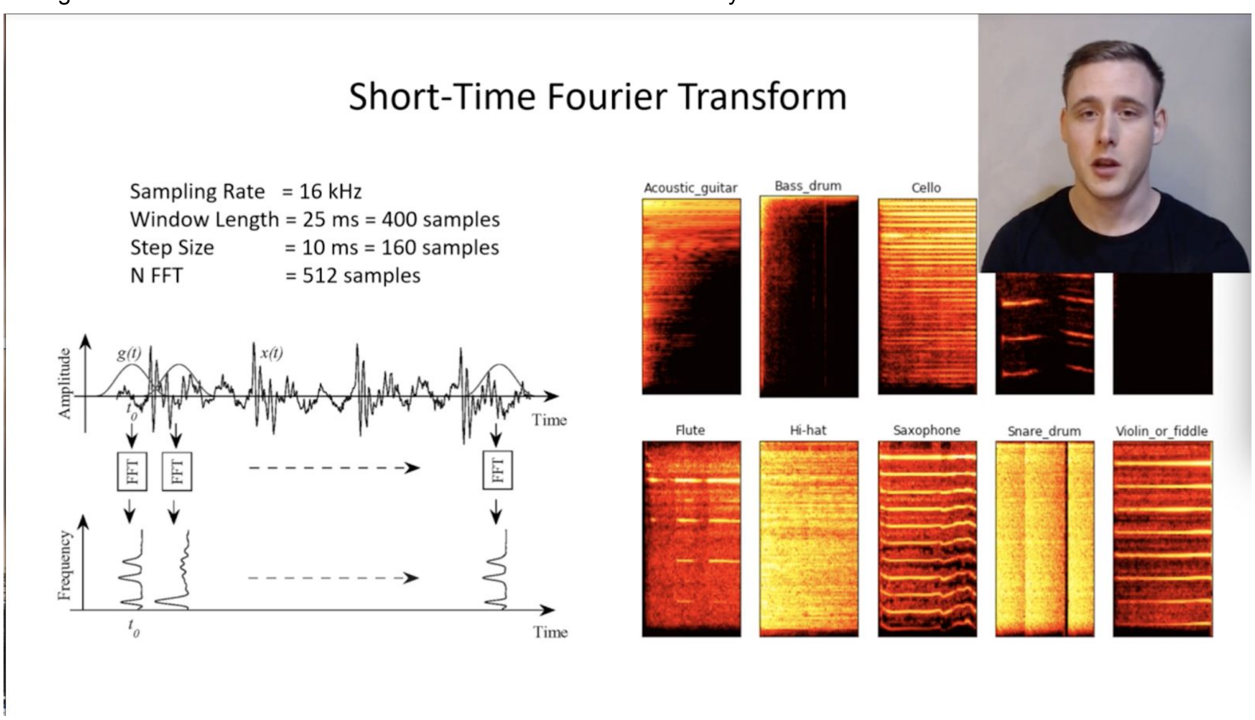
What does data look like: Import data from sensor
Sensor has bit depth (microphone, 16) 2^16 integer values
Express data in another format, we do a fast fourier transform
Construct a periodogram(Magnitude vs Freq, power spectral density estimate
for frequency bands)



Audio is typically recorded at 44.1kHz
Highest frequency we can represent from our environment. - Nyquist frequency (22 kHz)
Half of sampling frequency.
Cannot pick and represent any signal above Nyquist frequency
Most change happens at low frequency in audio. So we downsample our audio.
16kHz --> 8kHz
Spectogram



Periodogram stacked together over time. right next to each other.
Short Time Fourier Transform
Taking a small moment in time of the audio and assume it's stationery.



When we look at plots of stacked periodic data over time, we can see contours of different audio signals. Short Time Fourier Transform may be a good place to start classifying audio samples, as itt presents discrete samples of the audio signal over time. But it's possible to go further and make these samples more robust for classification.

Mel Scale
Humans can tell difference between low frequency values (10 and 100 Hz)
But once we get to higher frequency (15kHz) Humans can't tell tthe difference
Idea behind this: we don't care about difference in large freq but about
differences in low frequence (freq humans consider important)
Create a filter band over the power spectral density (periodogram)
Discretet Cosine Transform : Low pass filter for different energies, try to
remove high frequency by compacting information down
to lower frequencies.
Creates final feature: Mel Cepstrum Coefficients: ends up keeping low frequency

This was feature engineering on audio data.
http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/