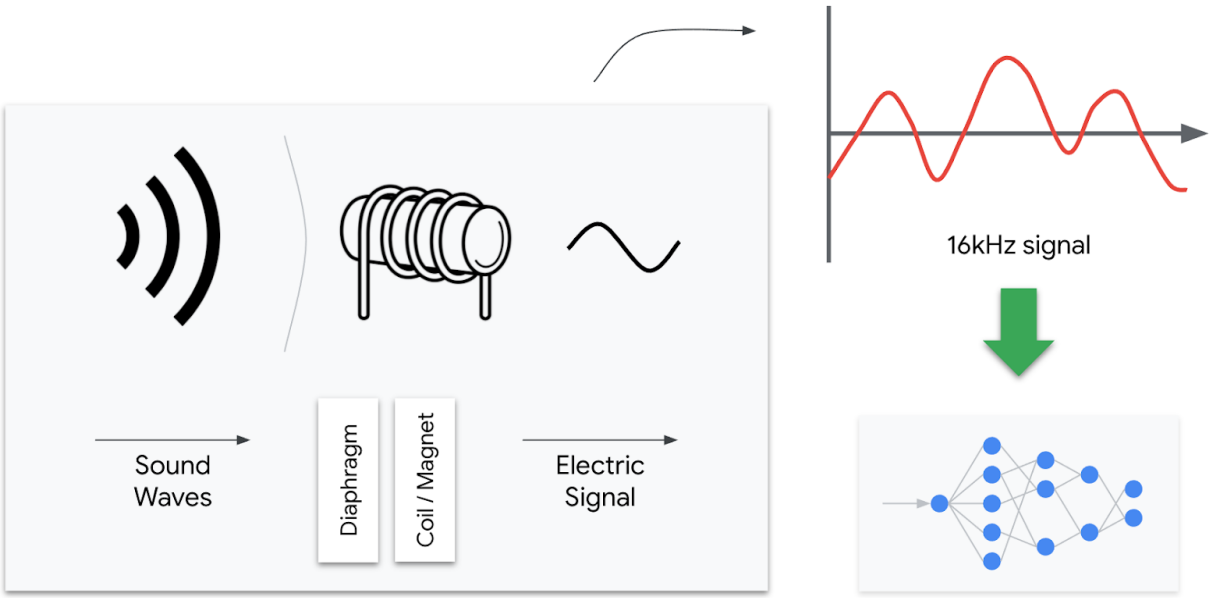
Feature Engineering for KWS

About This Reading

Keyword spotting (KWS) is one of the archetypal examples of TinyML in action. As we have discussed previously, the workflow consists of three stages: input, preprocessing, and output. Each of these stages is chained together in a data pipeline to output a label prediction for a specific audio signal. We will talk about each of the stages involved in the KWS workflow and delve into some of the specifics.

Input Signal Capture

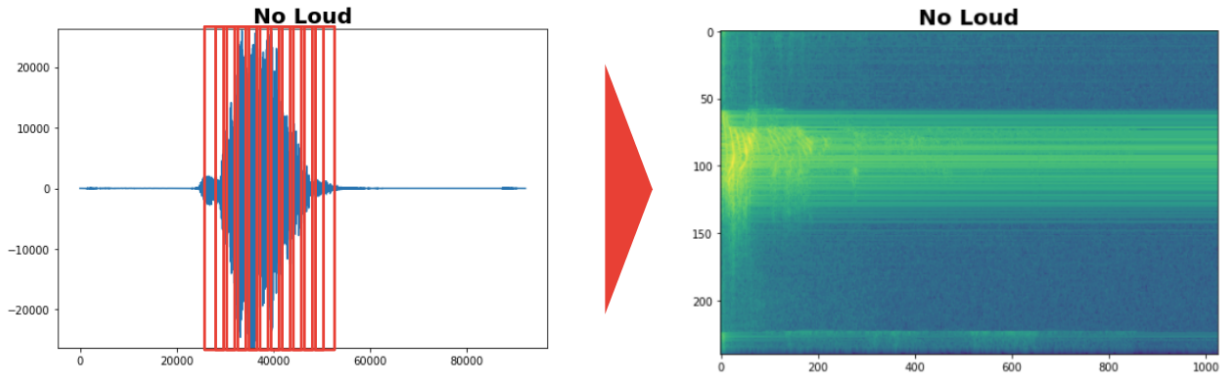
The input signal is obtained from an audio file or in real-time from an audio microphone. The conversion of sound from an audio microphone into a digital signal is relatively simple. The diaphragm in an audio microphone vibrates in the presence of sound, converting mechanical wave energy into an AC voltage signal (the method depends on the type of microphone, e.g., condenser vs. dynamic microphones). This AC signal is then converted into a digital signal. Thus, an audio microphone is essentially a transducer to convert sound into a voltage.



Beyond the microphone itself, there is a lot more going on in the input stage. Aspects such as the window length, window step, and the downsampling rate are important hyperparameters. Once determined, data is selected by “dragging” a window across the audio file and sending the data within this frame to the pre-processing stage.

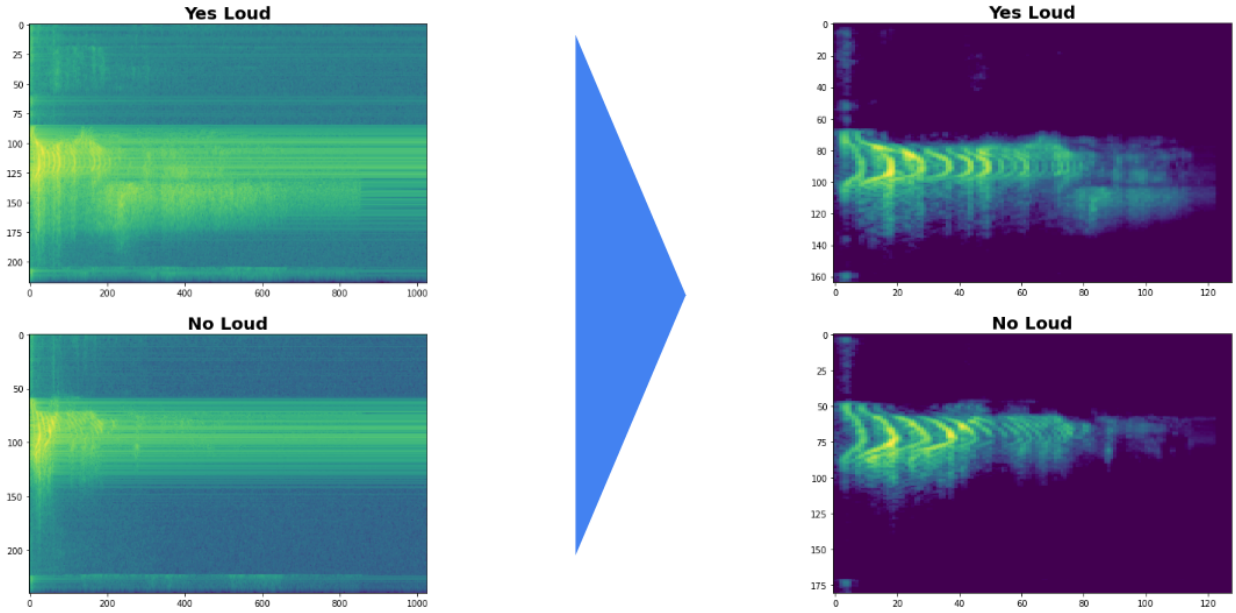
Pre-processing Blocks

During the pre-processing stage for KWS, the audio inputs are translated into a visual data format known as a **spectrogram** (as shown below). This is an image that reflects which frequencies were contained in the audio for each segment of time. Working with image data is easier than time-series data, and so converting the audio time-series to the spectrogram allows for more convenient analysis using a convolutional neural network.



As you might have expected, there are a wealth of hyperparameters that come with this preprocessing stage. Most notably, the frame length, frame stride, and frequency bands. These parameters tell us how long each segment of audio that we analyze will be, how many bands our frequency range will be split into, and how big the gaps between each window segment will be. Different values for these parameters will result in different spectrograms, which may be either easier or harder for our convolutional neural network to differentiate between. Ideally, we want to select values for these hyperparameters so that our network can easily differentiate between the different classes we wish to classify.

An alternative visual representation of audio data is the **Mel-frequency Cepstral Coefficients** (MFCCs) – as shown below. These coefficients are often used as features in speech recognition algorithms, and generally result in better performance compared to spectrograms. In human speech, a single MFCC usually corresponds with a particular phoneme, which is the smallest unit of sound that we can create. Syllables are created using a combination of phonemes, words and a combination of syllables. MFCCs are slightly less interpretable than a spectrogram, since they correspond with specific phonemes, but are more useful for extracting features by a convolutional neural network. Hyperparameters for MFCCs include the number of coefficients, frame length, frame stride, and FFT length.



The last audio-visual representation we will discuss for KWS is the **Mel-filter banks**. While MFCCs were popular for many years, nowadays Mel-filter banks are becoming increasingly popular. The two techniques are related, and, in fact, MFCCs can be derived from Mel-filter banks by the application of a discrete cosine transform. While the procedure is somewhat complex, the gist is that the audio signal is sent through a pre-emphasis filter, is sliced into overlapping frames, and then a window function is applied to each frame where a Fourier transform is then applied. From this, we can compute a power spectrum and subsequently compute the filter banks. For the interested reader, I recommend looking here for more information about filter banks.

Summary

In this reading, we have talked a bit about some of the feature engineering techniques used in KWS, mainly, the depiction of audio in a visual representation that can then be parsed by a convolutional neural network. The basis of these techniques: the spectrogram, MFCCs, and Mel-filter banks, lies in digital signal processing theory, and while not strictly necessary to understand to implement these for performing KWS, it is certainly helpful to understand the workings of each to determine which is the most suitable for a given application.