1. Data Cleaning

One of the biggest challenges of our model was that our prototype captures sound noise that surrounds the watch.

We tackled this challenge from both a mechanical and a post-processing approach.

In order to minimize the presence of the watch, we first have chosen a unidirectional microphone, point it out towards the watch so it would capture less surrounding noise.

From a post processing approach, we first tried to use built applications to perform the data cleaning, but we have achieved higher results once we built a personalized application.

* 1. Rattlesnake

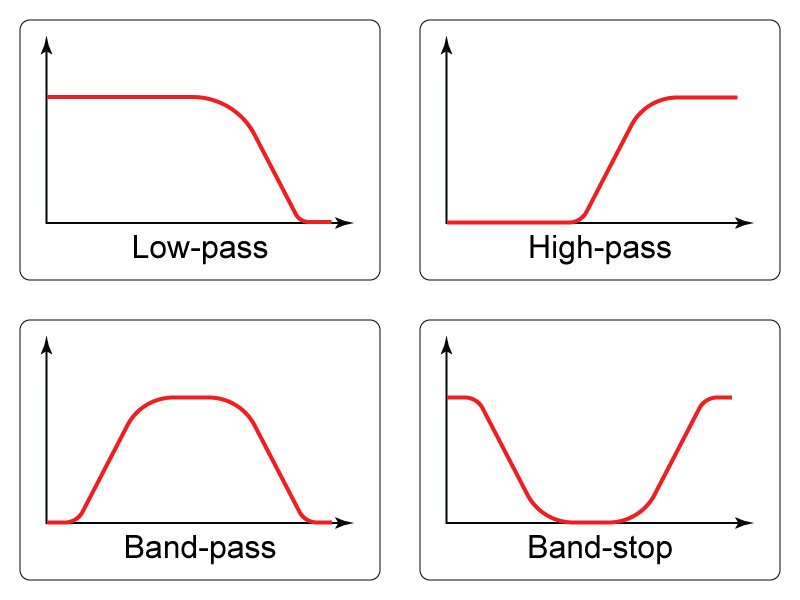
Rattlesnake is a Python application that does noise cancellation and more precisely *Active Noise Reduction (ANR).* ANR is the technique used for audio data cleaning by adding a sound specifically designed to cancel the noise.

Unfortunately, Rattlesnake considered that the watch was the noise, therefore we could not use it for our purpose

* 1. Filters

We built then our specific filters that would suppress certain ranges of frequencies. There are four types of filters:

* + - * Low-pass filters that allow only low frequencies to pass
      * High Pass filters that allow only high frequencies to pass
      * Band-pass filters that allow only a certain range to pass
      * Band-stop filters that blocks a certain range of frequencies to pass.



* 1. Exploratory Data Analysis

Exploratory Data analysis is the critical process of investigating the data to discover patterns and anomalies. In order to implement the right filter, we had to perform Exploratory Data Analysis (EDA). Our purpose is to be able to know what ranges of frequencies the sound emitting by the clock belongs to.

A spectrogram is **a visual way of representing the signal strength, or “loudness”**, of a signal over time at various frequencies present in a particular waveform. This would allow us to gain visual insight about our sound. The following figure is a Mel-spectrogram one of our recordings where the x-axis is the time axis, the y-axis represents the frequencies of the sounds, and the colors represent the level of loudness (i.e darker fragments of the picture refers to louder sounds).

We noticed in this process equally spaced lines in the higher frequencies who represented the ticks. Therefore, we have built a high-pass filter which reduced the noise, and offered acceptable output.

A picture containing text

Description automatically generated

A screenshot of a computer

Description automatically generated with low confidence

* 1. Data Separation

Audio Source Separation is the process of separating a mixture (e.g. a pop band recording) into isolated sounds from individual sources (e.g. just the lead vocals). As a future improvement, we can use this technique to isolate the sound of the watch from the surrounding noise.

In order to do that, we need to have a specific microphone that captures only the watch ideally surrounded by an isolating material, and another microphone outside the prototypes that captures the noise.

Using the built dataset, the model would learn to separate the sound of the watch from the noise and adapt the filters to the surrounding noise.

1. Capturing the metrics
   1. Chronometry

The first metrics that we want to capture are related to the chronometry.

We have used the Test and Measure Technology for Mechanical Watches report done by Witschi, that defines the metrics captured by a timegrapher.

* + 1. Rate deviation

To calculate the rate deviation the differences between the measured period and the nominal value are each averaged over the set measuring time.

A picture containing antenna

Description automatically generated

* + 1. Deviation Beat Error

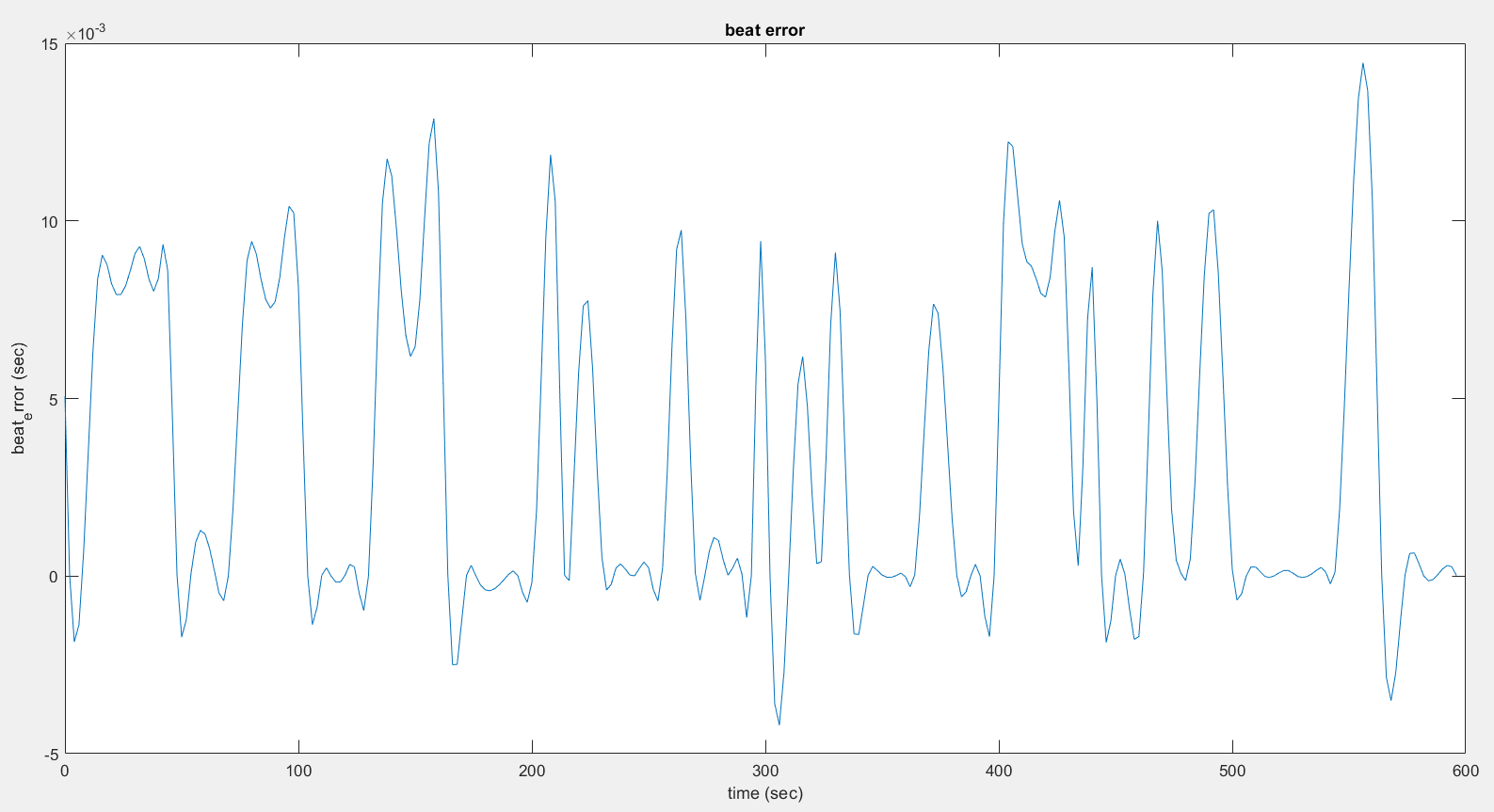
The deviation beat error captures asymmetrical oscillation of the balance wheel which is described by the rotating angle. When his metric is zero, it indicates perfect symmetrical oscillation.

A picture containing antenna

Description automatically generated

* + 1. Peak Detection

In order to capture both these metrics for our recordings, we need to accurately capture the peaks that occurs with the *tic-tac.* After using the high-pass filters, we capture the level of decibels performed by the chronometry and filter all sounds with higher decibels. This allows us to extract the metrics and show the following graphs.



* 1. Winding Speed

The winding speed is a second metric that we want to extract. When the watch recharges, it emits a specific sound that we will refer to as the *recharging sound*. Once we can detect the recharging sound, we can evaluate the winding mechanism.

In order to do that, we are going to build a Deep Learning model that trains to detect when the watch recharges. Our model will have an input one second of cleaned recording of the watch and will perform binary classification, meaning that it will output:

* + - * zero if the input does not contain the recharging sound
      * one if the input does contain the recharging sound
    1. Creating the dataset

The first challenge for this model was to create a *labeled* data: meaning recording of a watch when we know where it was recharged.

If we used a dataset from audio recordings, we would need a person to spend a day to listen to the watch and note when the watch does watch, which was not feasible.

Therefore, we have created our own dataset by recording the chronometry of the watch, the recharging sound and adding randomly the recharging sound at random times, which now allows us to know when the recharging occurs.

To avoid having a model that only recognizing the watch we have performed Data Augmentation.

Data augmentation in data analysis are techniques used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data. It helps reduce overfitting when training a machine learning model.

More specifically, we randomly injected added white Gaussian Noise, shifted frequencies making the pitch of the watch higher or lower and shifted the time of the recordings.

* + 1. Building the model

Despite that our dataset is an audio dataset, we tackle this problem from an image classification perspective. Every input of the model (one second of recording) is transformed to a spectrogram. We then build a Convolutional Neural Netowrk (CNN). Within Deep Learning, a Convolutional Neural Network or CNN is a type of artificial neural network, which is widely used for image/object recognition and classification.

We first split our data, into *train* data used to train the model, *val* to validate the parameters of our model and *test* to evaluate the accuracy of our model.

Then, we build our model and train it with *train.* The technical specifications of the model are clearly commented in the code.

* + 1. Evaluation

We achieve an accuracy of 83.6%. The first figures shows the evaluation of the accuracy throughout the epochs, and presents the following.

Chart, line chart

Description automatically generated

This figure presents the confusion matrix where we have 1050 values predicted correctly, 45 False negatives and 160 false positives.

Chart

Description automatically generated