Feature Extraction of the PeakAffect Data Set

# Introduction

**Overview (What is the thesis about?)**

PeakAffectDS is a dataset of physiological reactions to videos designed to provoke an emotional response. The purpose of this thesis is to take the raw data collected from these trials and extract a broad range of features. This will produce a dataset that balances human comprehension with machine learning potential, making accurate predictions of emotion in a way that makes the underlying reasoning understandable and reproducible.

**PeakAffectDS Background (What is PeakAffectDS?)**

The PeakAffectDS trials involved 51 different subjects. Each subject was shown 6 videos, each intended to provoke a different emotional response, with an additional 6 “neutral” videos shown in between each emotional video to return to baseline emotional status. The emotions provoked were “Calm”, “Happy”, “Sad”, “Angry”, “Fearful” and “Disgust”. During the viewing of these videos, an EMG of the cheek muscle (zygomaticus) and brow muscle (corrugator) were recorded, as well as the use of an ECG and respirator. These 4 physiological signals were recorded in mV for each of the 12 videos shown to each of the 51 subjects. Each emotional video was selected randomly from 3 different videos inducing the same emotion to prevent overfitting from one particular video.

# Cleaning Data

**Creating PSDs**

I began by downloading the raw data from PeakAffectDS and examining it. Steven Livingstone told me that the respiration and ECG data were already good, but the EMG data still needed to be properly sanitized. To look more closely at the EMGs, I created a distribution of frequencies using a power spectrum density graph (PSD). The PSD graphs were created using the “signal\_psd” function in the Neurokit2 package, drawing on the SciPy package, using the welch method for creating these PSD graphs by default.

**Notch Filtering**

Once we had PSD graphs, we started by examining the frequency distribution of subjects at random. We found that there was a large power spike at 50 Hz that needed to be filtered out. This was because the data was recorded in New Zealand where the power has a frequency of 50 Hz, which was interfering in the EMG recordings. To remove this interference, I applied a notch filter using the SciPy “iirnotch” function, which applies a notch filter on a signal given a specific frequency to filter and intensity (Q-factor). I tested this function on the data using different Q-factors to determine the best choice for filtering all the interference out, and we eventually settled on a Q-factor of 1 for 50 Hz. The power spike at 50 Hz was incredibly intense, requiring the use of such a filter despite how much it interfered with the surrounding frequencies. I then created one function that could apply this notch filter to every EMG signal in the dataset, and another that would plot the zygomaticus and corrugator PSD graphs of each subject’s physiology files side-by-side.

Once we could see the PSD graphs of each subject with the 50 Hz spike removed, we got a better sense of what the frequency distribution looked like. Despite the 50 Hz spike being removed, we noticed that all the signals appeared to have additional upper harmonic spikes every 100 Hz – at 150 Hz, 250 Hz, 350 Hz, 450 Hz, all the way to 950 Hz. To remove these as well, I modified the notch filter function to be able to apply multiple notch filters to every physiology file. I experimented with different Q-factors for the upper harmonics, looking at the resulting PSD graphs to arrive at a Q-factor small enough to remove the interference, but not so small that the surrounding frequencies would be significantly impacted. We eventually decided on a Q-factor of 25 for all the upper harmonics. We also re-evaluated our choice of 1 for the 50 Hz Q-factor due to how much it was affecting the surrounding frequencies. After experimentation, we came to decide on a Q-factor of 5 for 50 Hz, which was a better balance.

Once we had our main notch filters applied, we wanted to ensure there weren’t any additional interferences we had missed. We noticed an additional spike at 400 Hz common to each of the EMG signals in each of the subjects, so we added an extra notch filter at 400 Hz with a Q-factor of 25 for each of the subjects. To determine if there were any additional spikes we had missed, I created a function to find outliers in the PSD graphs, looking for points on the PSD graphs that lie unusually far from the rational function curve the graphs appeared to follow. After some development, the function drew our attention to subjects 1, 8 and 11. Subject 1 appeared to have a very messy EMG reading, however we determined that this was due to them being the first subject, the interference likely being from the EMG sensors having been put on improperly the first time the readings were taken. Subjects 8 and 11 appeared to have a strange spike at 317 Hz, not seen in other subjects. To clean this out, I modified the notch filter function to apply “special case” filters to specifically marked subjects and applied a notch filter of 317 Hz with a Q-factor of 25 to subjects 8 and 11 only.

**Bandpass Filter**

With all the notch filters applied, we then applied a bandpass filter. A bandpass filter is typically applied to remove low-frequency baseline drift, as well as high frequencies that can cause aliasing (<https://www1.udel.edu/biology/rosewc/kaap686/notes/EMG%20analysis.pdf>). For our application, we used a bandpass filter of 20 – 450 Hz. This was implemented using the SciPy “butter” function, using the Butterworth method to apply the filter.

**Full Wave Rectifier**

After the bandpass filter was applied, I applied a full wave rectifier to the data. This was done simply by taking the absolute value of the voltage of each EMG signal.

**Smoothing Functions**

Once the full wave rectifier was applied, we had to apply a smoothing function to the data to … **🡨 WHY?**. Literature indicated that the RMS method was the best choice for processing EMG signals **PROVIDE LINKS TO PROVE IT**, but to be sure we tried different methods and compared the results. I wrote code to perform RMS smoothing, boxcar smoothing, Gaussian smoothing and loess smoothing. I applied each of these smoothing functions to a physiology file and compared the results from each. The results seemed to confirm what the literature was saying – that RMS smoothing is the best method for EMG signals, so we decided on using RMS to smooth our signals.

# EMG Feature Extraction

**Toolboxes**

To perform the analysis, we wanted to make use of available toolboxes to extract features, starting with the EMG signals. We first looked at using more of NeuroKit2, which provided additional functions to process and analyze the signals, however the toolkit proved to be difficult to use if you did not follow its intended pipeline. We had done the cleaning ourselves and did not need to use the provided cleaning function, but the toolkit was designed in a way that each function is supposed to be used one after the other. In addition, closer inspection of the toolkit showed that it was mostly drawing on other Python libraries, serving more as a collection of useful functions from different sources rather than an original work. The analysis itself also seemed to be limited in the features it was able to extract, so we decided to look for a toolkit that could provide a more in-depth analysis.

Another toolkit we considered was PhysioData, a MATLAB toolkit. Like NeuroKit2, PhysioData is more difficult to use if you don’t follow their intended pipeline of starting with raw data, using the toolkit to perform the loading and cleaning, and finally extracting the features. However, it provides documentation for the format the data needs to be in to perform an analysis, allowing me to write a MATLAB script that converted our cleaned data into a form that could be analyzed. Looking at the features extracted by the toolkit, we noticed that there weren’t many in-depth analyses performed beyond simple summarization (min, max, mean, etc.). We also noticed that the toolkit was entirely GUI-based, with no option to use MATLAB scripts to automate the process.

We looked around for other toolboxes, but didn’t find anything that suited our needs, so we decided moving forward to perform the EMG analysis ourselves.

**Feature Extraction**

“There are three types of EMG features in different domains: time domain, frequency domain, and time-frequency domain” (<https://www.mdpi.com/1424-8220/13/9/12431>). In our analysis of the data, we focus on both time domain features and frequency features. We extracted the time domain features from the signals after they had been smoothed, and the spectral features from the data between the steps of applying a bandpass filter and applying a full-wave rectifier. This was to prevent the spectral results from being changed due to the smoothing **🡨 CONFIRM THIS WITH STEVEN LIVINGSTONE**. To extract frequency domain features, the data must be put into the frequency domain by creating a long-term average spectrum (LTAS). This represents a signal as a combination of frequencies of different power levels, showing the normalized power level of each frequency. Unlike a PSD graph, an LTAS is calculated using overlapping windows of the signal and averaging them, rather than using the entire signal (<https://www.researchgate.net/publication/317098414_Long-term_Average_Spectrum_in_Popular_Music_and_its_Relation_to_the_Level_of_the_Percussion>). The “signal\_psd” function used to create the PSD graphs by default uses the Welch method, which is used to calculate LTAS’s (<https://dsp.stackexchange.com/questions/14699/long-term-average-spectrum-of-large-batch-of-audio-files>), so we were able to proceed using this method to create our LTAS’s.

## EMG Basic Time Domain Extraction

We started by extracting basic summary features from the EMG signals. This includes the maximums, minimums, means and standard deviations of the signals. We were able to do these with basic Numpy functions. In addition, we used two less known statistical measures called “skewness” and “kurtosis” using SciPy.

Skewness describes the symmetry of a data set, being more skewed the less symmetrical the left and right of the median are (<https://www.investopedia.com/terms/s/skewness.asp>). It is calculated as follows:

Kurtosis describes how much data is in the tails in the bell curve of a distribution. (<https://www.investopedia.com/terms/k/kurtosis.asp>). It is calculated as follows, using the mean and standard deviation (<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6146874/>):

## EMG Manual Time Domain Extraction

After that, we extracted additional time-domain features that needed to be coded manually:

**Integrated EMG (IEMG)**

The Integrated EMG (IEMG) is the integral of the EMG signal – the area under the curve (<https://www.biopac.com/application/emg-electromyography/advanced-feature/automated-emg-analysis/>). This feature provides information about muscle activity (<https://lupinepublishers.com/biomedical-sciences-journal/fulltext/a-comprehensive-study-on-eMG-feature-extraction-and-classifiers.ID.000104.php>). The paper Surface Electromyography Signal Processing and Classification Techniques (<https://www.mdpi.com/1424-8220/13/9/12431>) describes this as the sum of absolute power values for the signal, however, this does not account for the sampling rate. The same signal under different sampling rates would have drastically different values if calculated this way. Given that the IEMG is the integral of the signal, it should be calculated as follows for a signal with power at frame and a sampling rate of :

**Mean Absolute Value (MAV)**

The Mean Absolute Value (MAV) is the mean of the absolute value of a signal (<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2881049/>). For a signal where is the power level at frame , the MAV is calculated as follows:

**Modified Mean Absolute Value 1 (MMAV1)**

The Modified Mean Absolute Value 1 (MMAV1) is a modified version of the MAV feature that gives more weight to the values in the middle of the signal. This is likely to reduce any additional error introduced at the beginning and end of the recording of the signal (<https://www.mdpi.com/1424-8220/13/9/12431>).

**Modified Mean Absolute Value 2 (MMAV2)**

The Modified Mean Absolute Value 2 (MMAV2) is another modified version of the MAV feature that uses a different set of weights (<https://www.mdpi.com/1424-8220/13/9/12431>).

**Variance (VAR)**

The Variance of a signal provides information about the power of the EMG signal (<https://lupinepublishers.com/biomedical-sciences-journal/fulltext/a-comprehensive-study-on-eMG-feature-extraction-and-classifiers.ID.000104.php>). It is calculated as follows:

**Simple Square Integral (SSI)**

The Simple Square Integral (SSI) provides information about the energy of the EMG signal (<https://lupinepublishers.com/biomedical-sciences-journal/fulltext/a-comprehensive-study-on-eMG-feature-extraction-and-classifiers.ID.000104.php>). The SSI is calculated by summing the square of the absolute value of each power value in the signal. Like the IEMG though, the SSI does not account for the sampling rate when calculating this feature. As such, the SSI calculation has been modified to the following to account for sampling rate:

**V-Order**

The V-Order of a signal is calculated using the operator, summing the values of the signal to the v-th power, and taking the v-th square root (<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2881049/>). A study has shown that the best choice for the v operator of an EMG signal is 2, which means the V-order is the same as the square root of the variance:

**Root Mean Square (RMS)**

The Root Mean Square (RMS) is a commonly extracted feature that provides information about the constant force, and non-fatiguing contractions of the muscles (<https://lupinepublishers.com/biomedical-sciences-journal/fulltext/a-comprehensive-study-on-eMG-feature-extraction-and-classifiers.ID.000104.php>). It is calculated as follows:

**Waveform Length (WL)**

The Waveform Length (WL) is the culmulative length of the signal over the period it has been recorded. The WL provides information about the amplitude, frequency and duration of the signal (<https://lupinepublishers.com/biomedical-sciences-journal/fulltext/a-comprehensive-study-on-eMG-feature-extraction-and-classifiers.ID.000104.php>). It is calculated as follows:

**Log-Detector (LOG)**

The Log-Detector of a signal provides an estimate of the force exerted by the muscle. It is calculated as follows (<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2881049/>):

**Willison Amplitude (WAMP)**

The Willison Amplitude (WAMP) is the number of times that an EMG amplitude’s change is greater than a defined threshold (<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2881049/>). Thresholds are commonly chosen in the 50 – 100 mV range. This could be affected by the sampling rate.

f(x) = {1 if x > ; 0 otherwise}

**Maximum Fractal Length (MFL)**

The Maximum Fractal Length (MFL) measures the activation of low-level muscle contractions (<https://www.mdpi.com/2079-3197/7/1/12>).

**Average Power (AP)**

The Average Power (AP) measures the energy distribution of a signal (<https://www.mdpi.com/2079-3197/7/1/12>). It is calculated by summing the power values squared and dividing by the signal length.

**Spectral Flux**

The spectral flux measures the change in the spectrums of two different signals (<https://www.sciencedirect.com/science/article/abs/pii/B9780080993881000042>). This is done by taking two frames of a signal, computing their normalized spectrum graphs, and finding the squared difference between them for each frequency. In our application however, we instead chose to compute the spectral of each emotional recording in relation to the neutral baseline recording taken in between viewings.

## EMG Frequency Domain Extraction

Finally, we extracted frequency domain features that we coded manually. It is worth noting that the LTAS is normalized after being calculated, but this does not interfere with the calculations below, as they are calculated as ratios and probabilities. Normalization makes sense in our application, as we are not trying to determine the intensity of the reaction, but the type of reaction. We want our features to distinguish between the different shapes and makeups of the frequencies of the emotions the subjects experience.

**Twitch Ratio**

The alpha ratio is a ratio between low-frequency energy and high-frequency energy, as separated by a threshold frequency selected from the distribution (The Geneva minimalistic acoustic parameter set, p.g. 199). This measure is useful for voice signals but does not have an equivalent for EMG signals. Thus, we have created a new measure that functions similarly that we have named the “twitch ratio”. Rather than dividing the signal by voice frequency, we divide it by muscle frequency. Human muscles can be classified as fast-twitch muscles, or slow-twitch muscles. Fast-twitch muscles are quicker and more powerful, slow-twitch muscles are slower but can be sustained longer. Fast-twitch muscles fire at higher frequencies, while slow-twitch muscles fire at lower frequencies (<https://blog.nasm.org/fitness/fast-twitch-vs-slow-twitch>). These muscles exist in the human face (<https://eu-ireland-custom-media-prod.s3.eu-west-1.amazonaws.com/UKMEAEU/eSample/9780323755115.pdf>), which is where the EMG readings are being recorded. Because of this, we propose finding the ratio of low-twitch muscle energy to high-twitch muscle energy. This is calculated the same way as the alpha ratio, except we use a threshold of 60Hz to divide the muscle types, with < 60Hz being from a slow-twitch muscle, and > 60Hz being from a fast-twitch muscle. This threshold value was chosen because in a study where slow-twitch and fast-twitch muscles were being exhausted and measured, the slow-twitch muscles had frequencies lower than 60Hz, while fast-twitch muscles had frequencies higher than 60Hz (<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7242282/>). It is calculated by dividing the sum of the power in the slow-twitch region by the sum of the power in the fast-twitch region.

**Twitch Index**

The Hammarberg index is a ratio comparing the power spike in the low-frequency region with the highest power spike in the high-frequency region, using the same threshold division as the alpha ratio (The Geneva minimalistic acoustic parameter set, p.g. 199). Following the same logic as the “twitch ratio”, we have replicated the Hammarberg Index using the same 60Hz threshold, calling this new metric the “twitch index”. It is calculated by dividing the maximum power level of the slow-twitch region by the maximum power level in the fast-twitch region.

**Twitch Slope**

The spectral slope is calculated by finding the linear least squares approximation of an LTAS (The Geneva Minimalistic Acoustic Parameter Set, p.g. 199). Spectral slopes are calculated for both the low and high frequency regions, as divided by the threshold used for the alpha ratio. Following the same logic as for the previous “twitch” metrics, we created the “twitch slope”, working the same as the spectral slope, but using the same 60Hz threshold. A twitch slope was calculated for both the slow-twitch region and the fast-twitch region.

For these twitch-related measures, it is important to note that depending on lifestyle, it is possible for your muscles to change from fast-twitch to slow-twitch, or from slow-twitch to fast-twitch. Power exercises will convert more of your muscles to fast-twitch muscles, while endurance exercises will convert more of your muscles to slow-twitch muscles (<https://blog.nasm.org/fitness/fast-twitch-vs-slow-twitch>). For these measurements, we have assumed that this factor is negligible, as regular intentional exercise of facial muscles is very unlikely. Additionally, as a person ages, there is a decline in the number of fast-twitch muscles, the impact of which is not currently being accounted for and warrants further study as these new measurements are investigated further.

**Spectral Centroid (SC)**

The spectral centroid is the “center of mass” of a signal that has had a Fourier transform applied, such as the LTAS (<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6806592/>). In this article, the spectral centroid was taken from a signal that had a discrete Fourier transform (DFT) applied, the same as in the LTAS’s created here. Thus, we can apply this metric to the LTAS’s we have created. The spectral centroid is calculated by summing the product of the power level of frequency k with k, and dividing by the sum of the power levels:

**Spectral Flatness (SF)**

The spectral flatness (SF) measures the noisiness of the magnitude spectrum (LTAS) (<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6146874/>).

**Spectral Spread (SS)**

The spectral spread (SS), also called the instantaneous bandwidth, measures the standard deviation around the spectral centroid (<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6146874/>).

**Spectral Decrease (SDec)**

The spectral decrease is the decrease of the slope of the spectrum with respect to its frequency (<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6146874/>). This is simply the slope of the LTAS.

**Spectral Entropy**

The spectral entropy is the Shannon entropy of the spectrum (<https://pubs.aip.org/asa/jasa/article/141/2/EL127/989855/Power-spectral-entropy-as-an-information-theoretic>). For this measure to be calculated, the spectrum must first be converted to a probability function by dividing each value by its sum.

**Spectral Rolloff**

The spectral rolloff point is the frequency of the power spectrum such that 85% of the total spectral energy lies below it (<https://musicinformationretrieval.com/spectral_features.html>). The actual threshold percentage can be set; however, the literature seems to suggest 85%

**Spectral Bandwidth (SBW)**

The spectral bandwidth is the difference between the upper and lower frequencies in the frequency band (<https://analyticsindiamag.com/a-tutorial-on-spectral-feature-extraction-for-audio-analytics/>). The signals oscillate around the centroid, so the bandwidth measures the standard deviation around the centroid. The parameter p can be adjusted to different values, but a value of 2 will produce the standard deviation around the centroid (<https://musicinformationretrieval.com/spectral_features.html>).

# Feature Reduction

After we had decided on all the features we could extract, we wanted to reduce the number of features. This is to improve the quality of the data, reducing the dimensionality by eliminating redundant or irrelevant data.

* WAMP usually uses a threshold of above 50mV, and our data doesn’t really go above that.
* V-Order is just the square root of the variance.
* We don’t have a threshold for looking at spectral flux.
* MMAV2 was created just for the article it was described in but is not adequately described and can’t be implemented.