The Journal of Open Source Software

EMGFlow: A Python package for preprocessing and feature extraction of electromyographic signals

William L. Conley [1¶] **and Steven R. Livingstone** [1]

1 Department of Computer Science, Ontario Tech University, Oshawa, Canada ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Fabian-Robert Stöter

Reviewers:

- @wbaccinelli
- @samiralavi

Submitted: 19 July 2024
Published: unpublished

Summary

Surface electromyography (sEMG) is increasingly used to study human physiology and behaviour, spurred by advances in deep learning and wearable sensors. Here, we introduce *EMGFlow*, an open-source Python package that streamlines preprocessing and feature extraction for sEMG signals. Tailored for batch processing, *EMGFlow* handles large datasets typical in machine learning, extracting a comprehensive set of 33 statistical features across time and frequency domains. The package supports flexible file selection with regular expressions and uses Pandas DataFrames end-to-end to facilitate interoperability. An interactive dashboard visualises signals at each preprocessing stage to aid user decisions. *EMGFlow* is distributed under the GNU General Public License v3.0 (GPL-3.0) and is available on PyPI. Documentation with guides, API references, and runnable examples is available at https://wiiison.github.io/EMGFlow-Python-Package/.

Statement of Need

Although several packages process physiological and neurological signals, support for sEMG has remained limited. Many lack a comprehensive feature set for sEMG, forcing researchers to use a patchwork of tools. Others focus on event detection with GUI-centric workflows that suit continuous recordings of a single participant, but complicate batch feature extraction common in machine learning (Abadi et al., 2015; Chen et al., 2022; Koelstra et al., 2012; Schmidt et al., 2018; Sharma et al., 2019; Zhang et al., 2016).

EMGFlow, a portmanteau of EMG and Workflow, fills this gap by providing a flexible pipeline for extracting a wide range of sEMG features, with a scalable design suited for large datasets. An overview of package metadata is presented in Table 1.

Metadata	Description
License	GPLv3
Implementation	Python >= 3.9
Code repository	https://github.com/Willson/EMGFlow-Python-Package
Documentation	https://wiiison.github.io/EMGFlow-Python-Package
PyPI installation	`pip install EMGFlow`

Table 1: *EMGFlow* package metadata.

Comparison to Other Packages

Compared to existing toolkits, *EMGFlow* provides a broader, sEMG-specific library of 33 features (Bizzego et al., 2019; Bota et al., 2024; Makowski et al., 2021; Sjak-Shie, n.d.;

Soleymani et al., 2017). Its dashboard visualises batch-processed files rather than single recordings, enabling inspection of preprocessing effects across datasets (Gabrieli et al., 2020). Adjustable filters and smoothing support international mains standards (50 vs 60 Hz), a subtle detail some packages omit.

Features

A Simplified Workflow

Extracting features from large datasets is fundamental in machine learning and quantitative analysis. *EMGFlow* supports batch-processing, enabling fully or semi-automated treatment of sEMG recordings. Figure 1 outlines the pipeline.

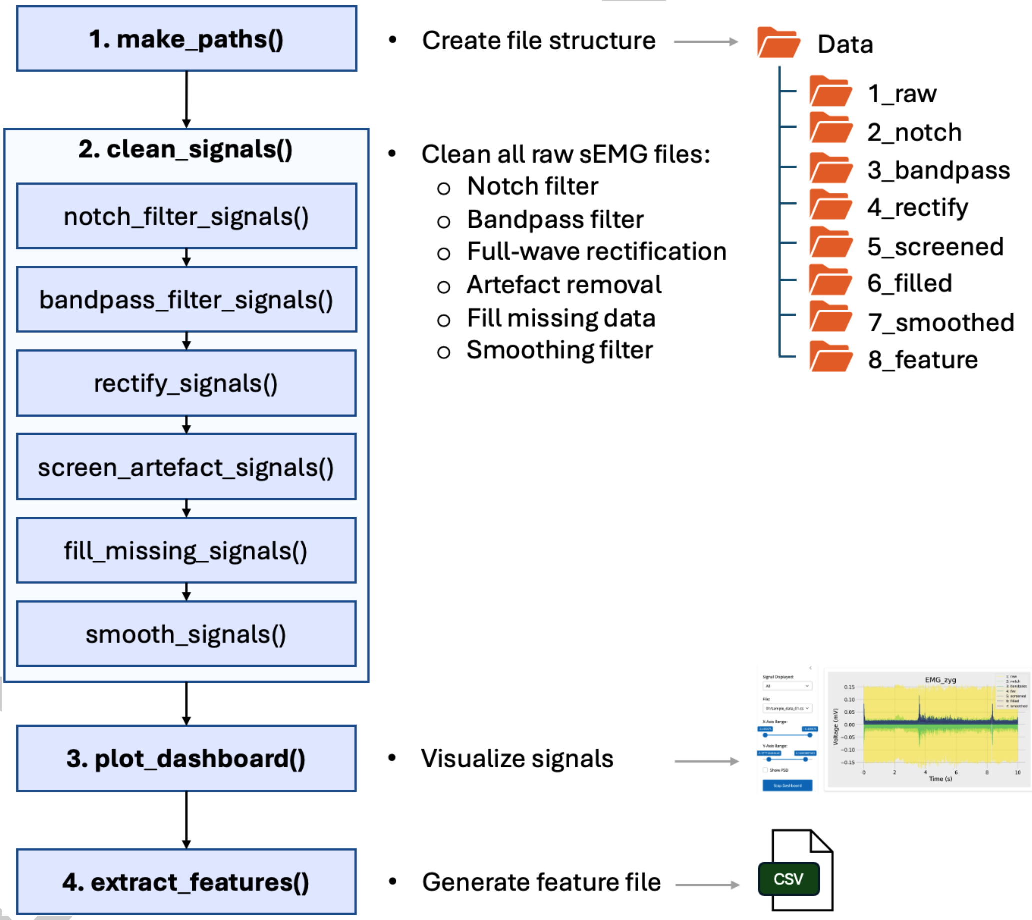

Figure 1: An overview of the processing pipeline.

Example 1 demonstrates end-to-end preprocessing and feature extraction. We create project paths with `make_paths()` and load bundled sample data with `make_sample_data()` (adapted from PeakAffectDS (Greene et al., 2022)). Next, we run automated preprocessing via `clean_signals()` using sensible, literature-based defaults, and then write a plaintext CSV of 33 features per file with `extract_features()`.

```python
# %% Example 1: Quick start (full pipeline)
import EMGFlow

# Create project paths
path_names = EMGFlow.make_paths()

# Load sample data
EMGFlow.make_sample_data(path_names)

# Preprocess signals
EMGFlow.clean_signals(path_names, sampling_rate=2000, notch_f0=50)

# Extract features to disk "Features.csv"
EMGFlow.extract_features(path_names, sampling_rate=2000)
```

Tailored Preprocessing

Example 2 shows how advanced users can tailor low-level preprocessing. After setup, Step 1 applies a notch filter to remove AC mains interference. Most functions use common sense defaults, which can be modified task-wide or for select cases. For instance, the sample data were recorded in New Zealand (200-240 VAC 50Hz), so we set the notch frequency and quality factor accordingly.

```python
# %% Example 2: Tailored preprocessing
import EMGFlow

# Setup workspace
path_names = EMGFlow.make_paths()
EMGFlow.make_sample_data(path_names)

# Data sampling rate
sampling_rate = 2000

# Notch filter for mains hum (Hz, Q-score)
notch_main = [(50, 5)]

# Columns names containing sEMG (Zygomaticus major, Corrugator supercilii)
muscles = ['EMG_zyg', 'EMG_cor']

# Step 1. Apply notch filter to all files in 1_raw, writing output to 2_notch
EMGFlow.notch_filter_signals(path_names['raw'], path_names['notch'],
                             muscles, sampling_rate, notch_main)
```

EMGFlow preserves the raw directory structure and mirrors it at each pipeline stage. All preprocessing functions accept an optional regular expression to target specific files. In Step 1b, we apply an additional notch filter at 150 Hz (the 3rd harmonic) only to files in subfolder `/01`.

```python
# Custom notch settings
notch_custom = [(150, 25)]
path_pattern = '^01/'

# Step 1b. Apply custom notch filter all to files in subfolder "/01"
EMGFlow.notch_filter_signals(path_names['notch'], path_names['notch'],
                             muscles, sampling_rate, notch_custom,
                             expression=path_pattern)
```

Interference Attenuation

Surface EMG is susceptible to multiple sources of interference that affect the signal with distinct spectral signatures (Boyer et al., 2023). Band-pass filtering is typically performed in Step 2 to isolate the frequency spectrum of human muscle activity. Common passbands are 10-500 Hz (Livingstone et al., 2016; McManus et al., 2020; Sato et al., 2021; Tamietto et al., 2009), though precise edges vary by domain (Abadi et al., 2015). Step 3 performs full-wave rectification, converting negative values to positive (Dakin et al., 2014; Rutkowska et al., 2024).

```
# Passband edges (low, high)
passband_edges = [20, 450]

# Step 2. Apply band-pass filter
EMGFlow.bandpass_filter_signals(path_names['notch'], path_names['bandpass'],
                                muscles, sampling_rate, passband_edges)

# Step 3. Apply full-wave rectifier
EMGFlow.rectify_signals(path_names['bandpass'], path_names['fwr'], muscles)
```

Signal artefacts are another source of contamination and span a diverse range of phenomenon including thermal noise, eyeblinks, and random noise bursts (Boyer et al., 2023). These can be mitigated with `screen_artefacts()`, which applies a Hampel filter (default), or Wiener filter, both reported as robust denoisers (Allen, 2009; Bhowmik et al., 2017; Jarrah et al., 2022). Because artefact profiles vary across projects, we recommend visual inspectection with the interactive dashboard to tune `n_sigma` (Hampel) and `window_ms` (Bhowmik et al., 2017; Pearson et al., 2016). In Step 4 we target `/02/sample_data_04.csv` which contains an artificial, band-limited noise pulse, and copy other files forward untouched.

```
screen_pattern = r'^02/sample_data_04\.csv$'

# Step 4. Apply Hampel artefact filter to 02/sample_data_04.csv
EMGFlow.screen_artefact_signals(path_names['fwr'], path_names['screened'],
                                muscles, sampling_rate,
                                expression=screen_pattern, copy_unmatched=True)
```

Missing data consisting of brief gaps or `NaN`s can be filled with `fill_missing_signals()`, which defaults to Piecewise Cubic Hermite Interpolating Polynomial (`method=pchip`). PCHIP is shape-preserving, monotonicity-respecting, and avoids overshoot - properites desirable for sEMG (SciPy Community, 2025). Cubic spline is also available (Shin et al., 2021). In Step 5, we address artificially injected gaps with PCHIP.

In Step 6, optional smoothing removes residual high-frequency noise before feature extraction. The default smoother RMS, equal to the square root of the total power, estimates signal amplitude and is commonly used in sEMG (McManus et al., 2020). Boxcar, Gaussian, and LOESS alternatives are also provided.

```
# Step 5. Fill missing data
EMGFlow.fill_missing_signals(path_names['screened'], path_names['filled'],
                             muscles, sampling_rate)

# Step 6. Apply smoothing filter
EMGFlow.smooth_signals(path_names['filled'], path_names['smooth'],
                       muscles, sampling_rate)
```

The Journal of Open Source Software

An Interactive Dashboard

EMGFlow includes a Shiny dashboard for visualising preprocessing effects. Pipeline steps can be overlaid or shown individually, and files are selected from a drop-down menu. A checkbox toggles between a time-domain amplitude view and a spectral view that displays the Power Spectral Density (PSD). The amplitude view exposes transients and drift, guiding selection of passband edges and confirming that filtering preserves waveform shape. The PSD highlights mains peaks and harmonics, guiding the choice of notch parameters (f0, Q). Below we generate a dashboard for the Zygomaticus major channel. When we have finished inspecting the signals, we click 'Stop Dashboard' to shut down the dashboard server and end the interactive session so that the analysis pipeline can proceed.

```python
# Column and measurement units to plot
show_muscle = 'EMG_zyg'
units = 'mV'

# Plot data for the "EMG_zyg" column
EMGFlow.plot_dashboard(path_names, show_muscle, sampling_rate, units)
```

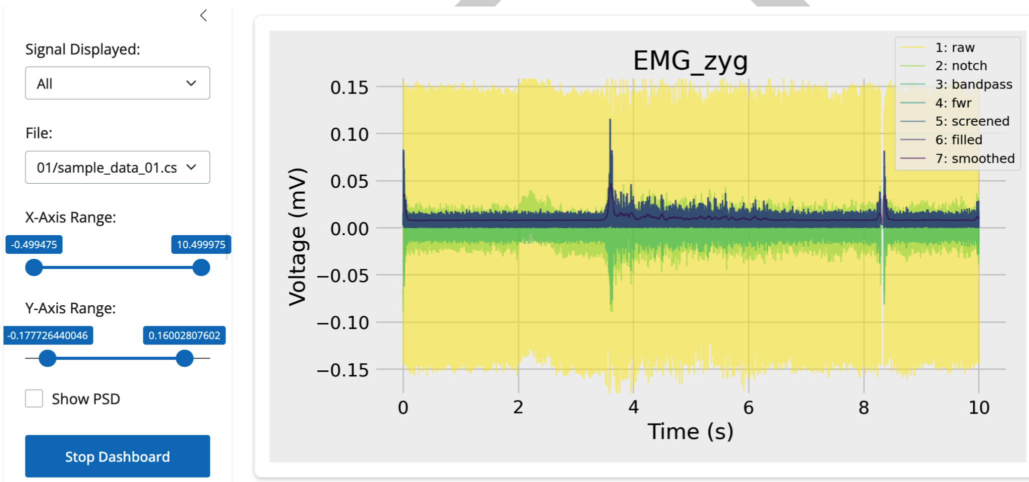

Figure 2: *EMGFlow*'s interactive dashboard visualizing effects of different preprocessing steps on batch processed files.

An Extensive Feature Library

After preprocessing, files are ready for feature extraction. Surface EMG records voltage differences at the skin arising from the summed motor-unit action potentials (Fridlund & Cacioppo, 1986), yielding an interference signal whose amplitude (time domain) and spectrum (frequency domain) reflect motor-unit recruitment, discharge rates, and muscle-fiber conduction velocity (De Luca, 2008; McManus et al., 2020). *EMGFlow* extracts 33 features across time and frequency domains, as listed in Table 2.

Domain	Feature
Temporal	minV, maxV, meanV, stdV, skewV, kurtosisV, maxF, IEMG, MAV, MMAV1, MMAV2, SSI, VAR, VOrder, RMS, WL, WAMP, LOG
Spectral	MFL, AP, SpecFlux, MDF, MNF, TwitchRatio, TwitchIndex, TwitchSlope, SC, SF, SS, SDec, SEntropy, SRoll, SBW

Table 2: Features extracted from sEMG signals.

We conclude Example 2 by extracting features, previewing the first rows, and outputting package metadata.

```
# Step 7. Extract features and save results in "Features.csv"
df = EMGFlow.extract_features(path_names, muscles, sampling_rate)

# Inspect features
df.round(4).head()

"""
                 File_Path  EMG_zyg_Min  ...  EMG_cor_SB  EMG_cor_Spectral_PCT_Missing
0  01/sample_data_01.csv       0.0031  ...    543.1803                        0.0050
1  01/sample_data_02.csv       0.0050  ...    346.9988                        0.0002
2  02/sample_data_03.csv       0.0001  ...   2183.3999                        0.0153
3  02/sample_data_04.csv       0.0024  ...   1051.9444                        0.0000

[4 rows x 71 columns]
"""

# Get package version
EMGFlow.package_version()

"""
EMGFlow 1.1.2
"""

# Get package citation
# EMGFlow.package_citation()
```

Temporal Feature Extraction

The set of 18 time-domain features include statistical moments (mean, variance, skew, kurtosis) and sEMG-specific measures. Examples include Willison amplitude, a proxy for motor unit firing that counts threshold crossings, and log-detector, an estimator of muscle force (Tkach et al., 2010). Time-domain features can be computed after the first three preprocessing steps (notch, band-pass, rectify); Steps 4-6 are optional.

Spectral Feature Extraction

The 15 frequency-domain features characterise power-spectrum shape and distribution. Median frequency (Phinyomark et al., 2009) tracks changes in conduction velocity and is used in muscle fatigue assessments (Boxtel et al., 1983; Lindstrom et al., 1977; McManus et al., 2020). Standard measures include spectral centroid, flatness, entropy, and roll-off. We also introduce Twitch Ratio, adapted from speech analysis (Eyben et al., 2016), defined as the ratio of upper- to lower-band energy with a 60 Hz boundary between slow- and fast-twitch muscles fibres (Hegedus et al., 2020).

Spectral features are computed by converting the Step 2 band-limited signal into a PSD. To avoid discarding otherwise valid Welch frames due to isolated dropouts, we perform constrained interpolation for micro-gaps <5 samples (2.5–5 ms at 1–2 kHz) and leave longer gaps as NaN so affected frames are rejected (Jas et al., 2017). This limits interpolation bias, which increases with gap size and density (Clifford & Tarassenko, 2005; Munteanu et al., 2016). We do not apply Steps 3–6 before PSD: rectification is non-linear and distorts spectra (Farina et al., 2013; McClelland et al., 2014; Neto & Christou, 2010); artefact-replacement filters can violate stationarity assumptions for FFT-based PSD; and smoothing suppresses high-frequency content. We estimate PSD with Welch's method using Hann windows, 50% overlap, and rejection of segments with remaining invalid samples, and mean averaging of retained spectra to form a long-term spectrum (Welch, 1967).

Missing Data Reporting

EMGFlow reports the percentage of missing data in the final temporal and spectral series as `_Temporal_PCT_Missing` and `_Spectral_PCT_Missing` in the extracted feature DataFrame, enabling downstream exclusion critera where appropriate.

Documentation and Testing

The documentation site (https://wiiison.github.io/EMGFlow-Python-Package) is built with VitePress. It provides a Quick-Start, an example gallery from minimal to advanced pipelines, an API reference with executable snippets, and a detailed catalogue of all mathematical feature definitions. Mermaid diagrams give a high-level view of the module structure.

Code reliability is enforced via an automated *unittest* suite run on every commit via GitHub Actions. The same tests can be executed locally; instructions and examples are provided on the documentation site. This ensures that changes remain reliable across platforms.

Community Guidelines

Contributions are welcome via issues or pull requests. Suggestions for features, usage tips, and questions can also be raised through GitHub Discussions.

AI Usage Disclosure

- Source code: All EMGFlow source code and test cases were written manually by the authors.
- Manuscript: The authors used GPT-5 to edit a final draft of the manuscript for flow, tone, and grammatical correctness. The authors reviewed and edited the content as needed and take full responsibility for the content of the publication.
- Documentation: The documentation website was created manually by the authors. The authors used GPT-5 to edit the "About electromyography" page. The authors reviewed and edited the content of this page and take full responsibility for the content of the page. All remaining documentation content was written manually by the authors.

Acknowledgements

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC, #2023-03786) and the Faculty of Science, Ontario Tech University.

Author contributions

S.R.L. conceptualised the project. W.L.C. and S.R.L. designed the toolbox functionality. W.L.C. wrote the toolbox code and maintains the GitHub repository. W.L.C. and S.R.L. maintain the documentation website. S.R.L prepared manuscript figures; W.L.C. prepared repository and documentation figures. S.R.L and W.L.C. prepared the manuscript and approved the final version.

References

Abadi, M. K., Subramanian, R., Kia, S. M., Avesani, P., Patras, I., & Sebe, N. (2015). DECAF: MEG-Based multimodal database for decoding affective physiological responses. *IEEE Transactions on Affective Computing*, *6*(3), 209–222. https://doi.org/10.1109/TAFFC.2015.2392932

Allen, D. P. (2009). A frequency domain hampel filter for blind rejection of sinusoidal interference from electromyograms. *Journal of Neuroscience Methods*, *177*(2), 303–310. https://doi.org/10.1016/j.jneumeth.2008.10.019

Bhowmik, S., Jelfs, B., Arjunan, S. P., & Kumar, D. K. (2017). Outlier removal in facial surface electromyography through hampel filtering technique. *2017 IEEE Life Sciences Conference (LSC)*, 258–261. https://doi.org/10.1109/LSC.2017.8268192

Bizzego, A., Battisti, A., Gabrieli, G., Esposito, G., & Furlanello, C. (2019). Pyphysio: A physiological signal processing library for data science approaches in physiology. *SoftwareX*, *10*, 100287. https://doi.org/10.1016/j.softx.2019.100287

Bota, P., Silva, R., Carreiras, C., Fred, A., & Silva, H. P. da. (2024). BioSPPy: A python toolbox for physiological signal processing. *SoftwareX*, *26*, 101712. https://doi.org/10.1016/j.softx.2024.101712

Boxtel, A. van, Goudswaard, P., Molen, G. M. van der, & Bosch, W. E. van den. (1983). Changes in electromyogram power spectra of facial and jaw-elevator muscles during fatigue. *Journal of Applied Physiology*, *54*(1), 51–58. https://doi.org/10.1152/jappl.1983.54.1.51

Boyer, M., Bouyer, L., Roy, J.-S., & Campeau-Lecours, A. (2023). Reducing noise, artifacts and interference in single-channel EMG signals: A review. *Sensors*, *23*(6). https://doi.org/10.3390/s23062927

Chen, J., Ro, T., & Zhu, Z. (2022). Emotion Recognition With Audio, Video, EEG, and EMG: A Dataset and Baseline Approaches. *IEEE Access*, *10*, 13229–13242. https://doi.org/10.1109/ACCESS.2022.3146729

Clifford, G. D., & Tarassenko, L. (2005). Quantifying errors in spectral estimates of HRV due to beat replacement and resampling. *IEEE Transactions on Biomedical Engineering*, *52*(4), 630–638. https://doi.org/10.1109/TBME.2005.844028

Dakin, C. J., Dalton, B. H., Luu, B. L., & Blouin, J.-S. (2014). Rectification is required to extract oscillatory envelope modulation from surface electromyographic signals. *Journal of Neurophysiology*, *112*(7), 1685–1691. https://doi.org/10.1152/jn.00296.2014

De Luca, C. J. (2008). A practicum on the use of sEMG signals in movement sciences. *Delsys Inc.*

Eyben, F., Scherer, K. R., Schuller, B. W., Sundberg, J., André, E., Busso, C., Devillers, L. Y., Epps, J., Laukka, P., Narayanan, S. S., & Truong, K. P. (2016). The Geneva Minimalistic Acoustic Parameter Set (GeMAPS) for Voice Research and Affective Computing. *IEEE Transactions on Affective Computing*, *7*(2), 190–202. https://doi.org/10.1109/TAFFC.2015.2457417

Farina, D., Negro, F., & Jiang, N. (2013). Identification of common synaptic inputs to motor neurons from the rectified electromyogram. *The Journal of Physiology*, *591*(10), 2403–2418. https://doi.org/10.1113/jphysiol.2012.246082

Fridlund, A. J., & Cacioppo, J. T. (1986). Guidelines for human electromyographic research. *Psychophysiology*, *23*(5), 567–589. https://doi.org/10.1111/j.1469-8986.1986.tb00676.x

Gabrieli, G., Azhari, A., & Esposito, G. (2020). PySiology: A python package for physiological feature extraction. In A. Esposito, M. Faundez-Zanuy, F. C. Morabito, & E. Pasero (Eds.), *Neural approaches to dynamics of signal exchanges* (pp. 395–402). Springer Singapore. https://doi.org/10.1007/978-981-13-8950-4_35

Greene, N., Livingstone, S. R., & Szymanski, L. (2022). *PeakAffectDS* (Version 1.0) [Data set]. Zenodo. https://doi.org/10.5281/zenodo.6403363

Hegedus, A., Trzaskoma, L., Soldos, P., Tuza, K., Katona, P., Greger, Z., Zsarnoczky-Dulhazi, F., & Kopper, B. (2020). Adaptation of fatigue affected changes in muscle EMG frequency characteristics for the determination of training load in physical therapy for cancer patients. *Pathology & Oncology Research*, *26*(2), 1129–1135. https://doi.org/10.1007/s12253-019-00668-3

Jarrah, Y. A., Asogbon, M. G., Samuel, O. W., Wang, X., Zhu, M., Nsugbe, E., Chen, S., & Li, G. (2022). High-density surface EMG signal quality enhancement via optimized filtering technique for amputees' motion intent characterization towards intuitive prostheses control. *Biomedical Signal Processing and Control*, *74*, 103497. https://doi.org/10.1016/j.bspc.2022.103497

Jas, M., Engemann, D. A., Bekhti, Y., Raimondo, F., & Gramfort, A. (2017). Autoreject: Automated artifact rejection for MEG and EEG data. *NeuroImage*, *159*, 417–429. https://doi.org/10.1016/j.neuroimage.2017.06.030

Koelstra, S., Muhl, C., Soleymani, M., Lee, J.-S., Yazdani, A., Ebrahimi, T., Pun, T., Nijholt, A., & Patras, I. (2012). DEAP: A database for emotion analysis using physiological signals. *IEEE Transactions on Affective Computing*, *3*(1), 18–31. https://doi.org/10.1109/T-AFFC.2011.15

Lindstrom, L., Kadefors, R., & Petersen, I. (1977). An electromyographic index for localized muscle fatigue. *Journal of Applied Physiology*, *43*(4), 750–754. https://doi.org/10.1152/jappl.1977.43.4.750

Livingstone, S. R., Vezer, E., McGarry, L. M., Lang, A. E., & Russo, F. A. (2016). Deficits in the mimicry of facial expressions in parkinson's disease. *Frontiers in Psychology*, *7*. https://doi.org/10.3389/fpsyg.2016.00780

Makowski, D., Pham, T., Lau, Z. J., Brammer, J. C., Lespinasse, F., Pham, H., Schölzel, C., & Chen, S. H. A. (2021). NeuroKit2: A Python toolbox for neurophysiological signal processing. *Behavior Research Methods*, *53*(4), 1689–1696. https://doi.org/10.3758/s13428-020-01516-y

McClelland, V. M., Cvetkovic, Z., & Mills, K. R. (2014). Inconsistent effects of EMG rectification on coherence analysis. *The Journal of Physiology*, *592*(1), 249–250. https://doi.org/10.1113/jphysiol.2013.265181

McManus, L., De Vito, G., & Lowery, M. M. (2020). Analysis and Biophysics of Surface EMG for Physiotherapists and Kinesiologists: Toward a Common Language With Rehabilitation Engineers. *Frontiers in Neurology*, *11*. https://doi.org/10.3389/fneur.2020.576729

Munteanu, C., Negrea, C., Echim, M., & Mursula, K. (2016). Effect of data gaps: Comparison of different spectral analysis methods. *Annales Geophysicae*, *34*(4), 437–449. https://doi.org/10.5194/angeo-34-437-2016

Neto, O. P., & Christou, E. A. (2010). Rectification of the EMG signal impairs the identification of oscillatory input to the muscle. *Journal of Neurophysiology*, *103*(2), 1093–1103. https://doi.org/10.1152/jn.00792.2009

Pearson, R. K., Neuvo, Y., Astola, J., & Gabbouj, M. (2016). Generalized hampel filters. *EURASIP Journal on Advances in Signal Processing*, *2016*(1), 87. https://doi.org/10.1186/s13634-016-0383-6

Phinyomark, A., Limsakul, C., & Phukpattaranont, P. (2009). *A novel feature extraction for robust EMG pattern recognition*. https://doi.org/10.48550/arXiv.0912.3973

Rutkowska, J. M., Ghilardi, T., Vacaru, S. V., Schaik, J. E. van, Meyer, M., Hunnius, S., & Oostenveld, R. (2024). Optimal processing of surface facial EMG to identify emotional expressions: A data-driven approach. *Behavior Research Methods*, *56*(7), 7331–7344. https://doi.org/10.3758/s13428-024-02421-4

Sato, W., Murata, K., Uraoka, Y., Shibata, K., Yoshikawa, S., & Furuta, M. (2021). Emotional valence sensing using a wearable facial EMG device. *Scientific Reports*, *11*(1), 5757. https://doi.org/10.1038/s41598-021-85163-z

Schmidt, P., Reiss, A., Duerichen, R., Marberger, C., & Van Laerhoven, K. (2018). Introducing WESAD, a multimodal dataset for wearable stress and affect detection. *Proceedings of the 20th ACM International Conference on Multimodal Interaction*, 400–408. https://doi.org/10.1145/3242969.3242985

SciPy Community. (2025). *PchipInterpolator — SciPy v1.16.2 manual*. https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.PchipInterpolator.html. https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.PchipInterpolator.html

Sharma, K., Castellini, C., Broek, E. L. van den, Albu-Schaeffer, A., & Schwenker, F. (2019). A dataset of continuous affect annotations and physiological signals for emotion analysis. *Scientific Data*, *6*(1), 196. https://doi.org/10.1038/s41597-019-0209-0

Shin, S. Y., Kim, Y., Jayaraman, A., & Park, H.-S. (2021). Relationship between gait quality measures and modular neuromuscular control parameters in chronic post-stroke individuals. *Journal of NeuroEngineering and Rehabilitation*, *18*(1), 58. https://doi.org/10.1186/s12984-021-00860-0

Sjak-Shie, E. E. (n.d.). PhysioData toolbox (version 0.7.0). In *PhysioData Toolbox*. Retrieved October 20, 2025, from https://physiodatatoolbox.leidenuniv.nl/

Soleymani, M., Villaro-Dixon, F., Pun, T., & Chanel, G. (2017). Toolbox for emotional feAture extraction from physiological signals (TEAP). *Frontiers in ICT*, *Volume 4 - 2017*. https://doi.org/10.3389/fict.2017.00001

Tamietto, M., Castelli, L., Vighetti, S., Perozzo, P., Geminiani, G., Weiskrantz, L., & Gelder, B. de. (2009). Unseen facial and bodily expressions trigger fast emotional reactions. *Proceedings of the National Academy of Sciences*, *106*(42), 17661–17666. https://doi.org/10.1073/pnas.0908994106

Tkach, D., Huang, H., & Kuiken, T. A. (2010). Study of stability of time-domain features for electromyographic pattern recognition. *Journal of NeuroEngineering and Rehabilitation*, *7*(1), 21. https://doi.org/10.1186/1743-0003-7-21

Welch, P. (1967). The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics*, *15*(2), 70–73. https://doi.org/10.1109/TAU.1967.1161901

Zhang, L., Walter, S., Ma, X., Werner, P., Al-Hamadi, A., Traue, H. C., & Gruss, S. (2016). "BioVid emo DB": A multimodal database for emotion analyses validated by subjective ratings. *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, 1–6. https://doi.org/10.1109/SSCI.2016.7849931