The Journal of Open Source Software

EMGFlow: A Python package for pre-processing 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: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished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 designed to simplify the preprocessing and feature extraction of sEMG signals. Tailored for batch processing, EMGFlow efficiently handles large datasets typical in machine learning contexts, extracting a comprehensive set of 33 statistical features across both time and frequency domains. The package integrates regular expression matching to support flexible file selection for preprocessing and feature extraction tasks. *EMGFlow* uses Pandas DataFrames throughout its workflow to facilitate interoperability with other data analysis tools. An interactive dashboard provides visual comparisons of signals at each preprocessing stage, enhancing user understanding and decision-making. *EMGFlow* is distributed under the GNU General Public License v3.0 (GPL-3.0) and can be installed directly from PyPI. A dedicated documentation site—complete with detailed guides, API references, and runnable examples—is available at https://wiiison.github.io/EMGFlow-Python-Package/.

Statement of Need

Although several packages exist for processing physiological and neurological signals, support for sEMG has remained limited. Many packages lack a comprehensive set of features that can be extracted from sEMG data, leaving researchers to use a patchwork of tools. Other packages are orientated around event detection in individual recordings and use a GUI-based workflow that requires more manual intervention. While this design works well for processing unedited continuous recordings of a single participant, it complicates the extraction of features from large datasets common to 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.

Comparison to Other Packages

Compared to existing toolkits, *EMGFlow* offers a more comprehensive set of 33 statistical features tailored specifically for sEMG signals (Bota et al., 2024; Makowski et al., 2021; Sjak-Shie, 2022; Soleymani et al., 2017). An interactive dashboard visualizes batch processed files rather than individual recordings, allowing the operator to view the effects of preprocessing across multiple datasets simultaneously (Gabrieli et al., 2020). Adjustable filter settings and smoothing functions accommodate different international standards (50 vs. 60 Hz mains AC), a critical detail overlooked by some packages.

Features

A Simplified Processing Pipeline

Extracting features from large datasets is fundamental in machine learning and quantitative analyses. *EMGFlow* supports batch-processing, enabling users to automate fully or semi-automate the treatment of sEMG recordings. Figure 1 illustrates the general pipeline. For demonstration, we use the built-in sample dataset from PeakAffectDS (Greene et al., 2022), containing data from two facial muscles: Zygomaticus major (Zyg) and Corrugator supercilii (Cor). We begin by creating file structures using `make_paths()`, then generating our sample data with `make_sample_data()`.

EMGFlow has been designed to allow researchers without deep signal-processing expertise to analyze sEMG data, while providing expert users flexability to control each step in the pipeline. Next, the `clean_signals()` function provides a high-level wrapper for automated preprocessing with sensible, literature-based defaults. Advanced users can customize individual processing steps as needed, as demonstrated below. Here, we apply a notch filter to remove AC mains interference, a common preliminary step in sEMG signal preprocessing.

```python
import EMGFlow

# Create processing file structure within '/Data'
path_names = EMGFlow.make_paths()

# Load sample data into Data/Raw
EMGFlow.make_sample_data(path_names)

# Sampling rate (in Hz)
sampling_rate = 2000

# Notch filter parameters (frequency, Q-factor)
notch_vals = [(50, 5)]

# Data columns for preprocessing
cols = ['EMG_zyg', 'EMG_cor']

# Apply notch filter (input path: /Raw, output path: /Notch)
EMGFlow.notch_filter_signals(path_names['Raw'], path_names['Notch'],
    sampling_rate, notch_vals, cols)
```

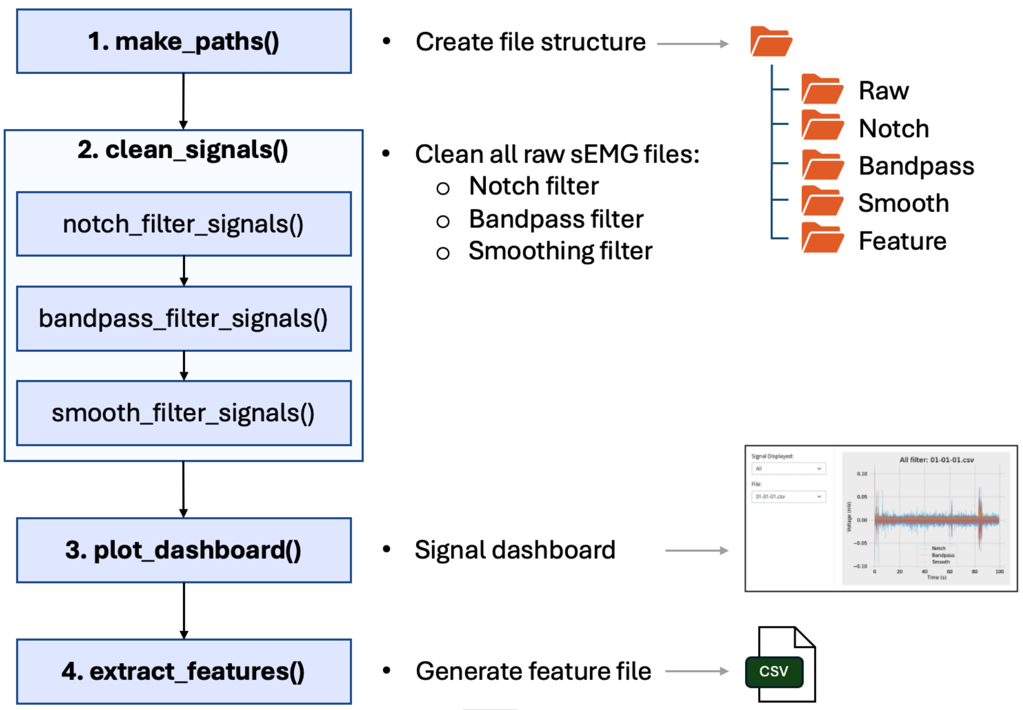

Figure 1: Overview of *EMGFlow*'s processing pipeline.

At each preprocessing step, *EMGFlow* mirrors the original input file structure, mapping its layout to the concomittant output folder. This approch maintains the researcher's data hierarchy, allowing for easy inspection of results at each stage of processing.

Advanced users can further customize preprocessing via regular expressions to selectively process subsets of files. Default parameters use canonical values widely recommended in the literature, and can be adjusted according to specific requirements. For example, in North America, mains electricity is nominally supplied at 120 VAC 60 Hz, while other countries may supply power at 200-240 VAC 50Hz. This variation in frequency requires different notch filter settings depending on where the data were recorded. Here we apply an additional notch filter to a subset of files belonging to participant 02, that exhibited noise at 150 Hz, the 3rd harmonic of the mains source.

```
# Secondary notch filter
notch_vals_subset = [(150, 25)]

# Match all .csv files within participant folder '02/'
participant_pattern = '02\/\w+.csv'

# Apply custom notch filter
EMGFlow.notch_filter_signals(path_names['Notch'], path_names['Notch'],
    sampling_rate, notch_vals_subset, cols, expression=participant_pattern)
```

Visualization of Preprocessing Stages

A bandpass filter typically follows notch filtering, as it isolates the frequency spectrum of human muscle activity. Signals are commonly filtered to the 10-500 Hz range (Livingstone et al., 2016; McManus et al., 2020; Sato et al., 2021; Tamietto et al., 2009), though precise filter corner frequencies vary by research domain and approach (Abadi et al., 2015). Next, data are smoothed to reduce high-frequency noise and outliers, enhancing temporal feature extraction accuracy. The default smoother is RMS, equal to the square root of the total power in the sEMG signal and commonly used to estimate signal amplitude (McManus et al., 2020).

Alternative smoothing methods include boxcar, Gaussian, and LOESS.

EMGFlow provides an interactive Shiny dashboard that allows users to visualize and contrast the effects of preprocessing on sEMG signals, as shown in Figure 2. Preprocessing stages can be displayed simultaneously or shown individually with options for Notch, Bandpass, and Smoothing steps. Users can select the file for visualization using the Files dropdown box. The dashboard is generated by extracting paths from the pipeline's file structure. Below, our example shows how signals are further bandpass filtered and smoothed, with results visualized using the dashboard.

```python
# Bandpass and smoothing filter parameters
band_low = 20
band_high = 140
smooth_window = 50

# Apply bandpass filter
EMGFlow.bandpass_filter_signals(path_names['Notch'], path_names['Bandpass'],
    sampling_rate, band_low, band_high, cols)

# Apply smoothing filter
EMGFlow.smooth_filter_signals(path_names['Bandpass'], path_names['Smooth'],
    smooth_window, cols)

# Data column and units for plotting
col = 'EMG_zyg'
units = 'mV'

# Plot all "EMG_zyg" data with interactive dashboard
EMGFlow.plot_dashboard(path_names, col, units)
```

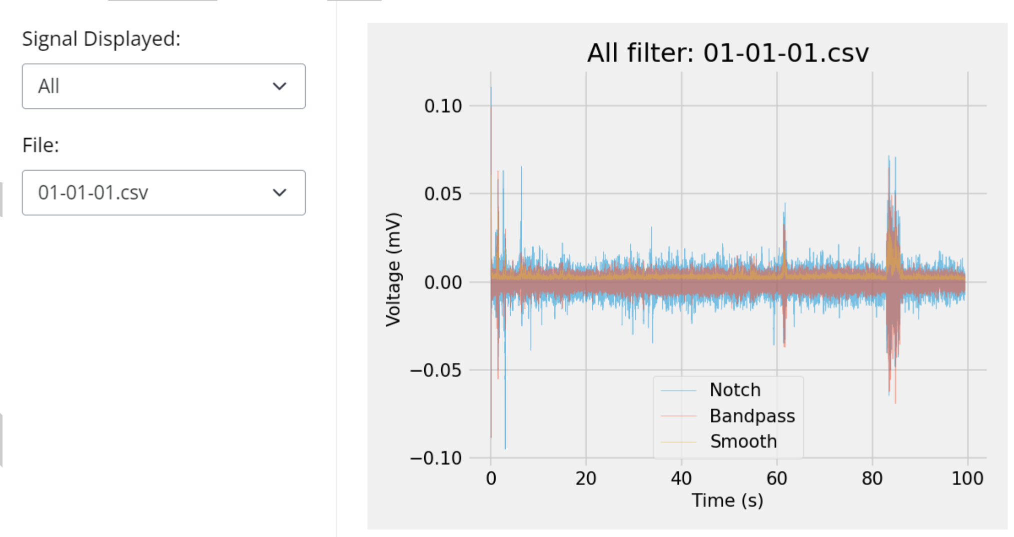

Figure 2: *EMGFlow*'s interactive dashboard shows the effects of preprocessing stages on batch processed files.

Feature Extraction

We begin with a review of surface electromyography as a recording instrument to better understand the range of features extracted by *EMGFlow*. Nearly all body movement occurs by muscle contraction. During contraction, nerve impulses sent from motoneurons cause muscle fibers innervated by the axon to discharge, creating a motor unit action potential (McManus et al., 2020). The speed at which action potentials propogate down the fibre is called muscle fiber conduction velocity. Each motor unit firing results in a force twitch. The superposition of these twiches over time produces a sustained force that enables functional muscle activity, such as lifting or smiling (De Luca, 2008).

Surface electromyography measures muscle activity by detecting the voltage differences produced as action potentials propagate through contracting muscle fibres. The resulting recordings form a voltage time-series that reflects the intensity and timing of muscle activation (Fridlund & Cacioppo, 1986). *EMGFlow* processes these voltage timeseries and extracts 33 features across time and frequency domains, as listed in Table 1. A key contribution of *EMGFlow* is that all features are computed natively with SciPy primitives; no external feature-extraction libraries are required (Virtanen et al., 2020). All computed features, along with their mathematical definitions, can be found in the Reference section of the Documentation website.

Eighteen time-domain features capture standanrd statistical moments, including mean, variance, skew, and kurtosis, along with sEMG-specific measures. These include features such as Willison amplitude, an indicator of motor unit firing calculated as the number of times the sEMG amplitude exceeds a threshold, and log-detector, an estimate of the exerted muscle force (Tkach et al., 2010). Fifteen frequency-domain features provide information on the shape and distribution of the signal's power spectrum. Measures such as median frequency (Phinyomark et al., 2009) provide insight into changes in muscle fibre conduction velocity and are used in the assessment of muscle fatigue (Lindstrom et al., 1977; McManus et al., 2020; Van Boxtel et al., 1983). Standard frequency measures include spectral centroid, flatness, entropy, and roll-off. An innovative feature is Twich Ratio, defined as the ratio of energy contained in the upper versus lower power spectrum, with a threshold of 60 Hz to delineate slow- and fast-twitch muscles fibres (Hegedus et al., 2020). Twitch ratio was adapted from Alpha Ratio in speech signal analysis (Eyben et al., 2016).

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 1: Features extracted from sEMG signals.

Our demonstration concludes with the extraction of features from our processed sEMG signal files. Features are summarized into a single CSV file, containing rows for each file analyzed, as shown below.

```
# Extract features and save results in "/Feature/Features.csv"
df = EMGFlow.extract_features(path_names, sampling_rate, cols)

# Print first few rows of extracted features table. The "File_ID" column
# contains the local path of files extracted, and the additional columns take
# the format "[Column name]_[Feature name]".
df.head()
"""
File_ID column contains

      File_ID                EMG_zyg_Min   ...  EMG_cor_Spec_Rolloff  EMG_cor_Spec_Bandwi
0  01/sample_data_01.csv     0.000826      ...  0.040222              1424.933862
```

```
1  01/sample_data_02.csv     0.000740     ...  0.019559              2651.987804
2  02/sample_data_03.csv     0.000780     ...  0.065183              2021.345274
3  02/sample_data_04.csv     0.000660     ...  0.087384              1755.834836

[4 rows x 61 columns]
"""
```

Documentation, Testing, and Availability

EMGFlow is supported by comprehensive documentation (https://wiiison.github.io/EMGFlow-Python-Package), generated with VitePress to provide a modern user experience (You & VitePress Contributors, n.d.). Documentation consists of (i) a Quick-Start tutorial for new users, (ii) an example gallery spanning three-line demonstrations to advanced processing pipelines, and (iii) an API reference annotated with executable code snippets, and mathematical definitions of all features. Package-level mind-maps, rendered with Mermaid.js (Sveidqvist & Mermaid Contributors, n.d.), provide a visual overview of the module hierarchy to assist user orientation.

Code reliability is maintained by an automated test suite, *unittest*, that executes on every GitHub commit via continuous-integration workflows. The complete source code is publicly available on GitHub under the GPL-3.0 licence - https://github.com/Willson/EMGFlow-Python-Package.

Community Guidelines

Contributions are encouraged via the project's issue tracker or pull requests. Suggestions for feature enhancements, tips, as well as general questions and concerns, can also be expressed through direct interaction with contributors and developers.

Declaration of Generative AI and AI-Assisted Technologies in the Writing Process

GPT-4.5 was used to edit the final manuscript draft. Authors carefully reviewed and finalized the content, and take full responsibility for the content of the publication.

Acknowledgements

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), (#2023-03786), and from 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. W.L.C. created and maintained the Github repository. S.R.L. and W.L.C. created the documentation website. W.L.C. and S.R.L. prepared figures for manuscript and Github repository. S.R.L and W.L.C. prepared the manuscript and approved the final version of the manuscript for submission.

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

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

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

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

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*. 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

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

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

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

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

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

Sjak-Shie. (2022). *PhysioData Toolbox* (Version 0.6.3). 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*, *4*. https://doi.org/10.3389/fict.2017.00001

Sveidqvist, K., & Mermaid Contributors, the. (n.d.). *Mermaid.js: Generation of diagrams and flowcharts from text*. https://mermaid-js.github.io.

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

Van Boxtel, A., Goudswaard, P., Van der Molen, G., & Van Den Bosch, W. (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

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., & others. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, *17*(3), 261–272.

You, E., & VitePress Contributors, the. (n.d.). *VitePress: Vite & vue powered static-site generator*. https://vitepress.dev.

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