Supplementary Information

Liquid Metal-based Sticky Conductor for Wearable and Real-Time Electromyogram Monitoring with Machine Learning Classification

Zixin Lin[1,#], Mingmei Luo[1,#], Jiayi Liang[2], Zijie Li[1], Yanting Lin[1], Xiaman Chen[2], Baozhu Chen[2], Liang Peng[2,*], Yongchang Ouyang[2,*], Lei Mou[1,*]

[1]The Key Laboratory of Advanced Interdisciplinary Studies, The First Affiliated Hospital of Guangzhou Medical University; School of Biomedical Engineering, Guangzhou Medical University, Yanjiang Road, Yuexiu District, Guangzhou, Guangdong 510120, P. R. China

[2]The Fifth Affiliated Hospital of Guangzhou Medical University, Department of Biotechnology, GMU-GIBH Joint School of Life Science, Guangzhou Medical University, Guangzhou, 511436, P. R. China

[*]Corresponding authors

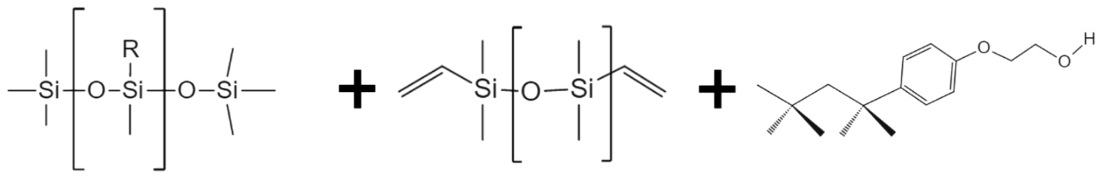
Liang Peng: pl_206@126.com

Yongchang Ouyang: ycouyang@gzhmu.edu.cn

Lei Mou: leimou@gzhmu.edu.cn

[#]These authors contributed equally.

Key works: Liquid Metal, Electromyogram, Soft electronics, Wearable Devices, Signal Processing



PDMS base PDMS crosslinker Triton-X

Figure S1. The molecular structure we have used for the modification of PDMS.

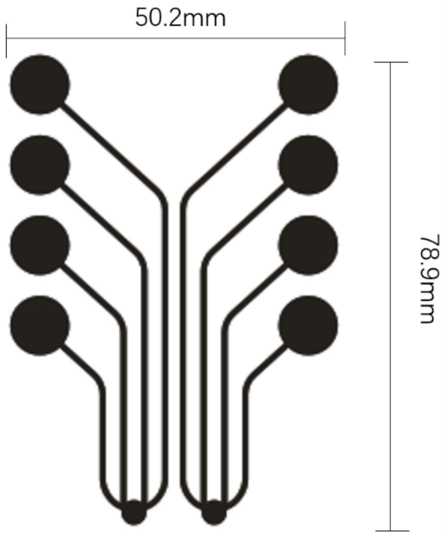


Figure S2. The design of the EMG electrode (50.2 mm wide, 78.9 mm long)

Figure S3. Photo of the fabricated LM ink.

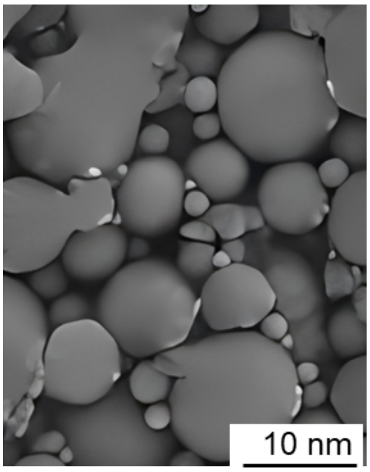


Figure S4. SEM images of the LM ink without AgNWs.

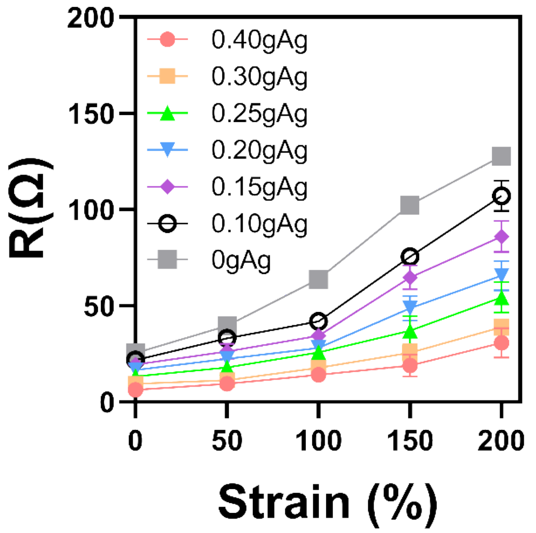


Figure S5. Resistance versus strain (%) for different concentrations of Ag (0.40g, 0.30g, 0.25g, 0.20g, 0.15g, 0.10g, 0g). Error bars indicate the standard deviation of the measured resistance change values, showing the variability and reliability of the measurements over multiple experiments. Each data point represents the average of multiple independent experiments. Different color markers correspond to different silver concentrations, indicating the repeatability and reproducibility of the results under the same experimental conditions.

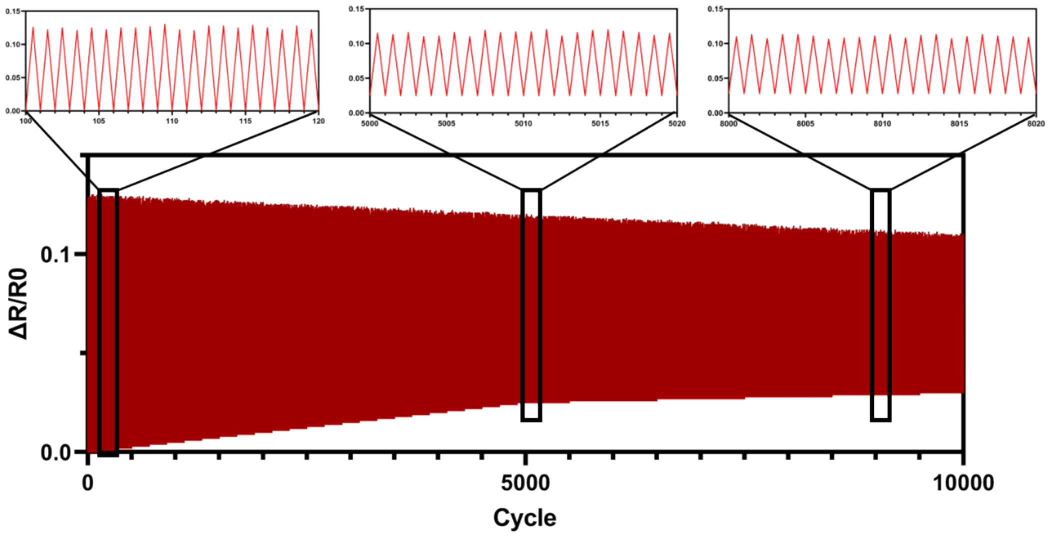
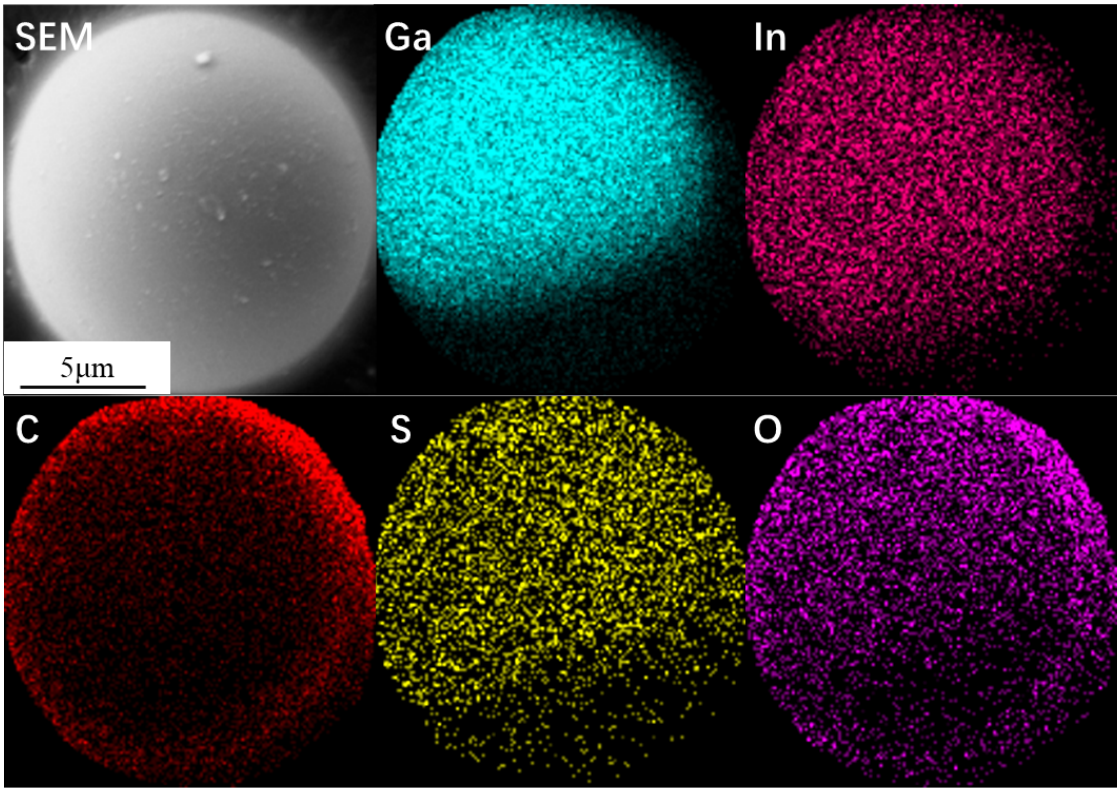


Figure S6. Stability data for resistance changes of the LM-PDMS electrode over 10,000 cycles

48


49 **Figure S7.** EDS images of LM-PDMS particles.

50

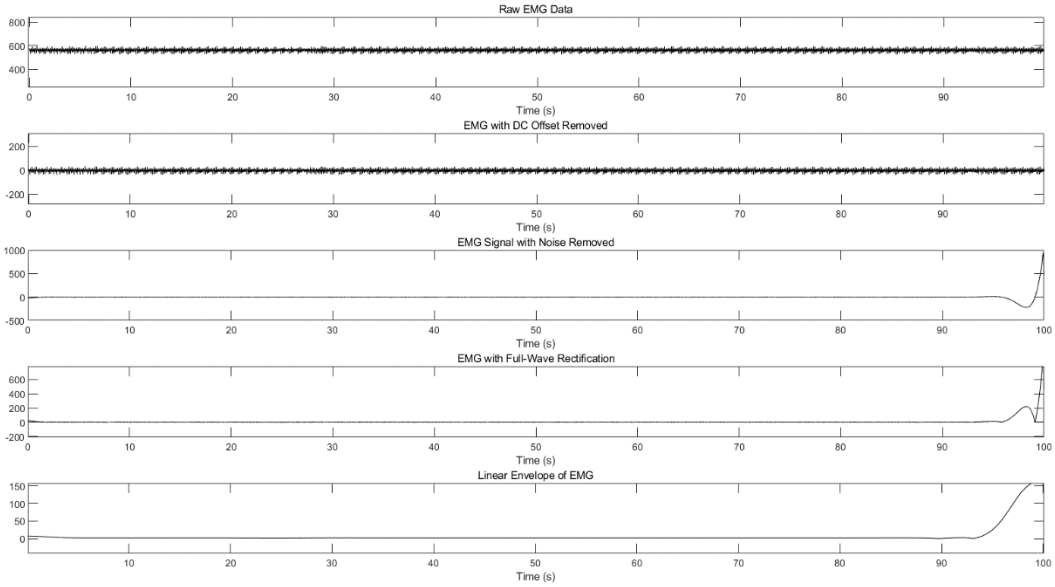


Figure S8. The raw EMG data in the resting state and its corresponding curve after multi-step processing.

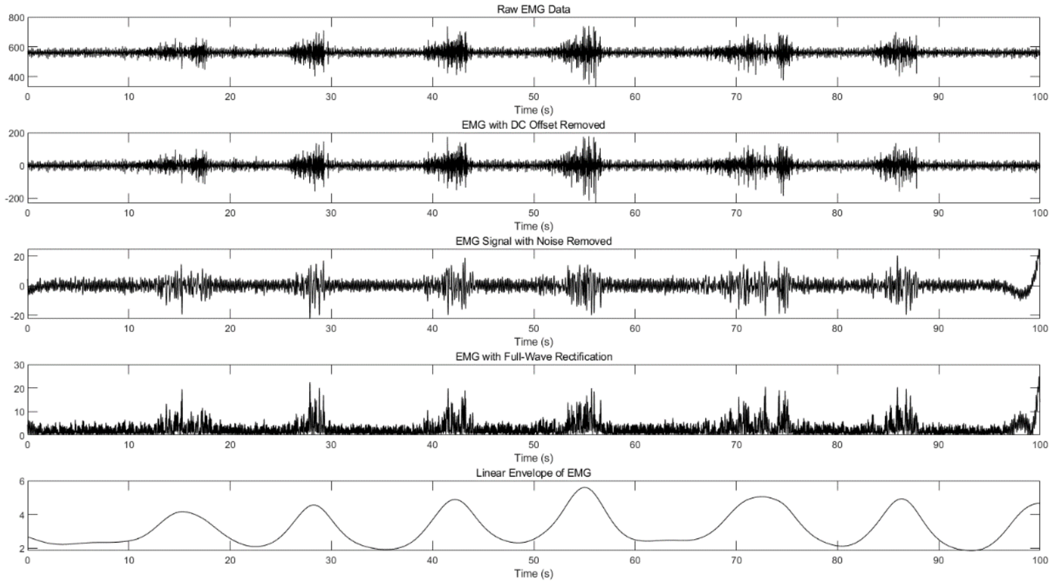


Figure S9. The raw EMG data from a clenched fist and the resultant curve following multi-step processing.

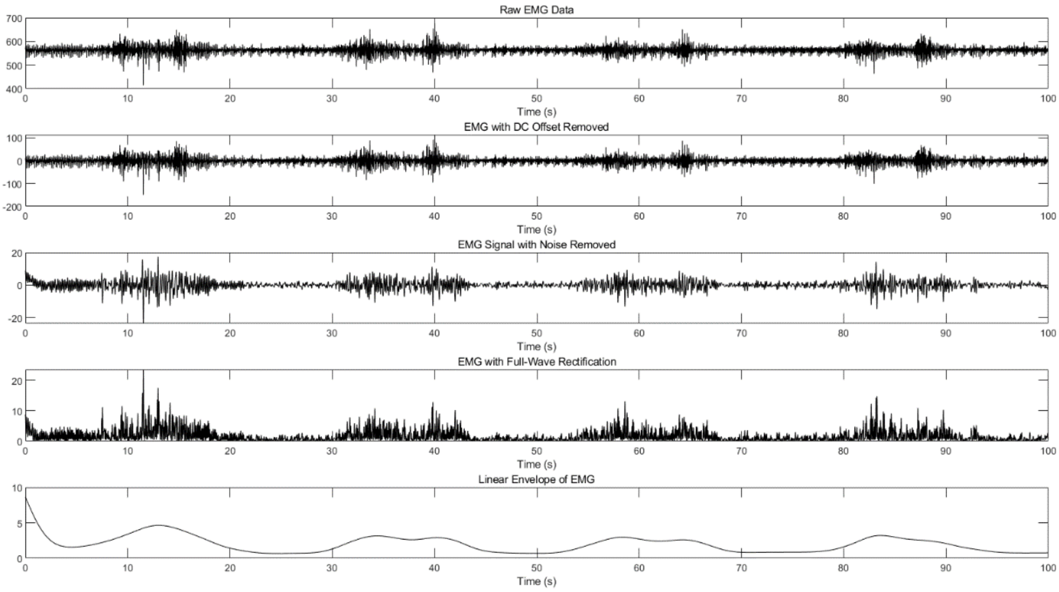


Figure S10. The raw EMG data from the forearm during wrist rotation and the plotted curve after multi-step processing.

Ref.	Stable Resistance Strain Range (%)
Our study	50
[1]	36
[2]	40
[3]	26
[4]	60
[5]	100
[6]	30
[7]	4
[8]	60

Table S1. Comparison of stable resistance strain ranges in different studies

1. Lee, H.; Lee, S.; Kim, J.; Jung, H.; Yoon, K. J.; Gandla, S.; Park, H.; Kim, S., *npj Flexible Electronics* **2023,** *7* (1), 20. DOI 10.1038/s41528-023-00246-3.
2. Zhao, Y.; Chen, C.; Lu, B.; Zhu, X.; Gu, G., *Advanced Functional Materials* **2024,** *34* (14), 2312480. DOI https://doi.org/10.1002/adfm.202312480.
3. Gong, Q.; Jiang, X.; Liu, Y.; Yu, M.; Hu, Y., *Advanced Electronic Materials* **2023,** *9* (9), 2200916. DOI https://doi.org/10.1002/aelm.202200916.
4. Zhao, Q.; Gribkova, E.; Shen, Y.; Cui, J.; Naughton, N.; Liu, L.; Seo, J.; Tong, B.; Gazzola, M.; Gillette, R.; Zhao, H., *Science Advances 10* (18), eadn7202. DOI 10.1126/sciadv.adn7202.
5. Wang, W.; Wang, S.; Rastak, R.; Ochiai, Y.; Niu, S.; Jiang, Y.; Arunachala, P. K.; Zheng, Y.; Xu, J.; Matsuhisa, N.; Yan, X.; Kwon, S.-K.; Miyakawa, M.; Zhang, Z.; Ning, R.; Foudeh, A. M.; Yun, Y.; Linder, C.; Tok, J. B. H.; Bao, Z., *Nature Electronics* **2021,** *4* (2), 143-150. DOI 10.1038/s41928-020-00525-1.
6. Zhang, L.; Kumar, K. S.; He, H.; Cai, C. J.; He, X.; Gao, H.; Yue, S.; Li, C.; Seet, R. C.-S.; Ren, H.; Ouyang, J., *Nature Communications* **2020,** *11* (1), 4683. DOI 10.1038/s41467-020-18503-8.
7. Kim, T.; Park, J.; Sohn, J.; Cho, D.; Jeon, S., *ACS Nano* **2016,** *10* (4), 4770-4778. DOI 10.1021/acsnano.6b01355.
8. Jang, K.-I.; Han, S. Y.; Xu, S.; Mathewson, K. E.; Zhang, Y.; Jeong, J.-W.; Kim, G.-T.; Webb, R. C.; Lee, J. W.; Dawidczyk, T. J.; Kim, R. H.; Song, Y. M.; Yeo, W.-H.; Kim, S.; Cheng, H.; Rhee, S. I.; Chung, J.; Kim, B.; Chung, H. U.; Lee, D.; Yang, Y.; Cho, M.; Gaspar, J. G.; Carbonari, R.; Fabiani, M.;

Gratton, G.; Huang, Y.; Rogers, J. A., *Nature Communications* **2014,** *5* (1), 4779. DOI 10.1038/ncomms5779

```
close all;
clear all;
clc
% Set sampling frequency and read EMG data
fs = 10000; % Sampling frequency (Hz)
EMG_DATA = xlsread('data.xlsx'); % Read the data from an Excel file
% Extract individual channels from the data
EMG_DATA_1 = EMG_DATA(:,1);
EMG_DATA_2 = EMG_DATA(:,2);
EMG_DATA_3 = EMG_DATA(:,3);
% Process each channel with the EMG processing function, including wavelet analysis
[emg_data_1, t, wResults1, waveletPower1] = processemg(EMG_DATA_1, fs, 6, 2);
[emg_data_2, t, wResults2, waveletPower2] = processemg(EMG_DATA_2, fs, 6, 2);
[emg_data_3, t, wResults3, waveletPower3] = processemg(EMG_DATA_3, fs, 6, 2);
% Normalize the processed EMG data based on the max contraction value
m_EMG_DATA_1 = max(emg_data_1);
perc_EMG_DATA_1 = (emg_data_1 / m_EMG_DATA_1) * 100;
m_EMG_DATA_2 = max(emg_data_2);
perc_EMG_DATA_2 = (emg_data_2 / m_EMG_DATA_2) * 100;
m_EMG_DATA_3 = max(emg_data_3);
perc_EMG_DATA_3 = (emg_data_3 / m_EMG_DATA_3) * 100;
% Create a new figure for EMG percentage of MVC plot
figure;
subplot(3,1,1);
plot(t, perc_EMG_DATA_1, 'black', 'LineWidth', 1.5);
xlabel('Time(s)');
ylabel('EMG (%MVC)');
legend('ARM Muscle in silent')
subplot(3,1,2);
plot(t, perc_EMG_DATA_2, 'black', 'LineWidth', 1.5);
xlabel('Time(s)');
ylabel('EMG (%MVC)');
legend('ARM Muscle with wrist twisting')
subplot(3,1,3);
plot(t, perc_EMG_DATA_3, 'black', 'LineWidth', 1.5);
xlabel('Time(s)');
ylabel('EMG (%MVC)');
title('EMG of ARM during Squat (%MVC)');
legend('ARM Muscle with fist up')
```

Code S1. The annotated MATLAB main function code for processing the forearm EMG signals.

```
function [output, t, waveletResult, waveletPower] = processemg(EMGRAW, fs, LOWPASSRATE, NUMPASSES)
  % Ensure NUMPASSES is reasonable, often 2 or 4.
  % Handle NaNs or Infinities here if they exist.
  EMGRAW(~isfinite(EMGRAW)) = 0;
  % Read data and create time vector
  t = linspace(0, (length(EMGRAW)-1)/fs, length(EMGRAW))*100;
  % Plot raw EMG data
  figure();
  subplot(5,1,1);
  plot(t,EMGRAW,'black');
  xlabel('Time (s)');
  title('Raw EMG Data');
  % Remove DC offset
  EMGDC = EMGRAW - mean(EMGRAW);
  subplot(5,1,2);
  plot(t,EMGDC,'black');
  xlabel('Time (s)');
  title('EMG with DC Offset Removed');
  % Filter the noise with bandpass filter
  Wn1 = 20/(fs/2);
  Wn2 = 600/(fs/2);
  [b,a] = butter(NUMPASSES,[Wn1 Wn2],'bandpass');
  EMGFILT = filtfilt(b,a,EMGDC);
  subplot(5,1,3);
  plot(t,EMGFILT,'black');
  xlabel('Time (s)');
  title('EMG Signal with Noise Removed');
  % Full-wave rectification
  EMGFWR = abs(EMGFILT);
  subplot(5,1,4);
  plot(t,EMGFWR,'black');
  xlabel('Time (s)');
  title('EMG with Full-Wave Rectification');
  % Wavelet decomposition
  waveletName = 'db9';
  level = wmaxlev(length(EMGFWR), waveletName);
  [C,L] = wavedec(EMGFWR, level, waveletName);
  % Store wavelet result
  waveletResult.cfs = C;
```

```
    waveletResult.lvl = L;
    waveletResult.detailCfs = cell(1, level);
    waveletResult.approxCfs = appcoef(C, L, waveletName);
    for i = 1:level
        waveletResult.detailCfs{i} = detcoef(C, L, i);
    end
    % Create wavelet power spectrum
    waveletPower = cell(1, level);
    for i = 1:level
        waveletPower{i} = abs(waveletResult.detailCfs{i}).^2;
    end
    % Linear Envelope
    Wn = LOWPASSRATE/(fs/4);
    [b,a] = butter(NUMPASSES, Wn, 'low');
    EMGLE = abs(filtfilt(b, a, EMGFWR));
    % Optionally remove the end spike
    subplot(5,1,5);
    plot(t,EMGLE,'black');
    xlabel('Time (s)');
    title('Linear Envelope of EMG');
    output = EMGLE; % Return the linear envelope result
end
```

Code S2. The MATLAB function code designed for sEMG signal processing, including removing DC offset, denoising through filtering, and extracting the linear envelope.

```python
import pandas as pd
import numpy as np
import scipy.fft as sp_fft
from scipy.signal import butter, filtfilt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from imblearn.over_sampling import SMOTE  # Import SMOTE for handling class imbalance
# Load the data
file_path = 'D:/matlab/Fist up.xlsx'
data = pd.read_excel(file_path)
# Preprocessing function
def preprocess_signals(data, lowcut=20, highcut=500, fs=1000, order=5):
    data_centered = data - data.mean()
    nyq = 0.5 * fs
    low = max(lowcut / nyq, 0.01)
    high = min(highcut / nyq, 0.99)
    b, a = butter(order, [low, high], btype='band')
    filtered_data = filtfilt(b, a, data_centered)
    return filtered_data
# Apply preprocessing
preprocessed_data = data.apply(preprocess_signals, axis=0)
def enhanced_features(data):
    data_array = data.values
    fft_values = sp_fft.fft(data_array)
    psd = np.abs(fft_values) ** 2
    normalized_psd = psd / np.sum(psd)
    mean_val = np.mean(data_array)
    std_val = np.std(data_array)
    # Flatten the feature array properly
    return np.concatenate([normalized_psd, [mean_val, std_val]])
# Apply enhanced feature extraction and collect features in a proper format
feature_list = []
for column in preprocessed_data.columns:
    features = enhanced_features(preprocessed_data[column])
    feature_list.append(features)
```

```
# Convert list of arrays into a 2D array
feature_matrix = np.array(feature_list)
# Prepare labels
labels = ['fist'] * 10 + ['wrist'] * 10
# Handling class imbalance
smote = SMOTE()
X_res, y_res = smote.fit_resample(feature_matrix, labels)
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_res, y_res, test_size=0.3, random_state=42)
# Classifier with hyperparameter tuning
model = RandomForestClassifier()
param_grid = {'n_estimators': [10, 50, 100], 'max_depth': [None, 10, 20, 30]}
grid_search = GridSearchCV(model, param_grid, cv=5)
grid_search.fit(X_train, y_train)
# Predict and evaluate
y_pred = grid_search.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
print("Best parameters:", grid_search.best_params_)
print("Accuracy:", accuracy)
print("Classification Report:\n", report)
```

Code S3. The Python code designed for EMG signal processing and classification using random forest.