EMG Project

Digital Signal Processing

Adham Ali - 900223243
Omar Saqr - 900223343
Saif Abdelfattah - 900225535

Submitted to Dr. Seif Eldawlatly
This paper is prepared for CSCE3611, section 01

Department of Computer Science and Engineering
The American University in Cairo
09/12/2024

Table of contents

Chapter 1

Introduction

This report describes the implementation of various methods for processing surface electromyographic (sEMG) signals and classifying the data. We use different feature extraction techniques, including time-domain and frequency-domain features, to classify the signals and determine the optimal hyperparameters for the classification model.

Chapter 2

Approach and Functions Overview

2.1 Importing Data

The function `import_mat_data` is responsible for loading the data from a `.mat` file. It extracts two key components: the EMG data and the movement stimulus.

```
def import_mat_data(file_path):
    loaded_data = scipy.io.loadmat(file_path)
    emg_data = loaded_data['emg']
    movement_stimulus = loaded_data['stimulus'].flatten()
    print(f"Muscle␣Data␣Shape:␣{emg_data.shape}")
    print(f"Movement␣Stimulus␣Shape:␣{movement_stimulus.shape}")
    return emg_data, movement_stimulus
```

Listing 2.1: Importing Data

- **EMG Data**: A 2D array containing the signals from multiple electrodes. - **Movement Stimulus**: A 1D array containing the movement labels associated with the EMG data. - The function also prints the shapes of the extracted data for verification.

2.2 High-pass Filtering

The function `apply_highpass_filter` applies a high-pass filter to the EMG signals. This filter is designed to remove low-frequency noise while retaining higher-frequency components of the signals.

```
def apply_highpass_filter(emg_data, cutoff_frequency=10, sampling_rate=100,
   filter_order=4):
    nyquist_frequency = 0.5 * sampling_rate
    normalized_cutoff = cutoff_frequency / nyquist_frequency
    filter_b, filter_a = butter(filter_order, normalized_cutoff, btype='high',
        analog=False)
    filtered_emg = filtfilt(filter_b, filter_a, emg_data, axis=0)
    return filtered_emg
```

Listing 2.2: Applying High-pass Filter

- **Cutoff Frequency**: The frequency below which the signals are attenuated. - **Sampling Rate**: The rate at which the data is sampled (100 Hz). - **Filter Order**: The order of the filter, controlling the steepness of the cutoff.

The function returns the filtered EMG data.

2.3 Feature Extraction (Time-domain)

The function `Get_timeD_Features` extracts time-domain features by flattening the signal from a single trial of a specific channel.

```
def Get_timeD_Features(ch_data):
    return ch_data.flatten()
```

Listing 2.3: Extracting Time-domain Features

The function takes the data of one channel, flattens it, and returns it as a 1D vector to be used for classification.

2.4 Feature Extraction (Frequency-domain)

The function `Get_freqD_Features` performs a Fourier Transform on the signal and returns the frequency-domain features.

```
def Get_freqD_Features(signal_data):
    FourierFt_res = np.fft.fft(signal_data, axis=0)
    FourierFt = np.abs(FourierFt_res)
    mid = len(FourierFt) // 2
    FourierFt = FourierFt[:mid]
    return FourierFt.flatten()
```

Listing 2.4: Extracting Frequency-domain Features

- The function performs a **Fourier Transform** on the signal to convert it to the frequency domain. - It returns the absolute values of the Fourier coefficients, truncated to half the spectrum (due to symmetry).

2.5 Splitting Trials Based on Stimulus Changes

The function `split_trials_by_stimulus` segments the EMG data into trials based on changes in the stimulus data.

```
def split_trials_by_stimulus(emg_data, stimulus_data, sampling_rate,
   duration_per_trial=5, excluded_stimulus=0):
    ...
    return trial_list, trial_labels
```

Listing 2.5: Splitting Trials by Stimulus

Purpose: Segments the EMG data into trials based on changes in the stimulus, where each trial represents one movement action.

Inputs:

- `emg_data`: The EMG data for the subject.
- `stimulus_data`: The stimulus labels, indicating the movements performed by the subject.
- `sampling_rate`: The sampling rate of the data.
- `duration_per_trial`: The duration of each trial in seconds.
- `excluded_stimulus`: A stimulus label to exclude from the trials (e.g., no movement).

Outputs:

- `trial_list`: A list of segmented trials.
- `trial_labels`: The labels (stimulus) corresponding to each trial.

Explanation: This function identifies the points where the stimulus changes, indicating a new trial. It then extracts the corresponding trial segments from the EMG data. Trials that have fewer than the required number of samples are discarded, and trials with certain stimulus labels can be excluded.

2.6 KNN Classification with Leave-One-Out Cross-Validation

The function `leave_one_out_knn_classification_method3` performs Leave-One-Out Cross-Validation (LOO-CV) using the KNN classifier.

```
def leave_one_out_knn_classification_method3(data_features, data_labels,
    k_range=range(1, 20)):
    ...
    return Kscores
```

Listing 2.6: Leave-One-Out KNN Classification

- **KNN Classifier**: The function evaluates the accuracy of different values of **K** (number of neighbors) in the KNN algorithm.

It returns a dictionary of accuracy scores for each value of K.

2.7 Finding the Best K

The function `find_optimal_k_and_accuracy` calculates the optimal value of **K** by averaging the accuracy scores for each **K** and selecting the one with the highest average accuracy.

```
def find_optimal_k_and_accuracy(Kscores):
    ...
    return optimal_k, best_avg_accuracy
```

Listing 2.7: Finding the Best K

This function helps to identify the best value of **K** for classification based on cross-validation results.

2.8 Plotting K vs Accuracy

The function `plot_k_vs_accuracy` generates a plot to visualize the relationship between the number of neighbors K and the classification accuracy.

```
def plot_k_vs_accuracy(Kscores, feature_description):
    ...
    plt.show()
```

Listing 2.8: Plotting K vs Accuracy

The plot helps to visualize how the KNN performance changes as the value of **K** is adjusted.

2.9 Method 1: Time-domain Feature Extraction

In this method, we extract time-domain features from the raw EMG signal data for each trial. The time-domain features are derived by flattening the signal for each channel. This transformation results in a 1D vector, which can then be used for machine learning classification.

2.9.1 Procedure

- The raw trials are first segmented into individual trials based on the stimulus data.
- Each channel's data is flattened into a 1D vector for each trial.
- The flattened features are then used to train and test the K-Nearest Neighbors (KNN) classifier.
- The performance of the classifier is evaluated using Leave-One-Out Cross-Validation (LOO-CV) across different values of K.

2.9.2 Code Explanation

The function `process_time_domain_1` handles the time-domain feature extraction and classification. It processes the data by extracting features for each channel and evaluating the classification accuracy for different values of K.

```
def process_time_domain_1(raw_trials, labels):
    num_ch = raw_trials[0].shape[1]
    k_values = range(1, 20)
    accuracy = []

    for ch in range(num_ch):
        ch_features = [trial[:, ch].flatten() for trial in raw_trials]
```

```
        accuracies_for_k = [leave_one_out_knn_classification(ch_features,
            labels, k) for k in k_values]
        accuracy.append(accuracies_for_k)

    return accuracy
```

Listing 2.9: Time-domain Feature Extraction

- `raw_trials`: The segmented EMG signal data.
- `labels`: The corresponding labels (stimuli) for each trial.
- For each channel in the raw trials, we extract features by flattening the data into a 1D array.
- The function then evaluates the classification accuracy using KNN for each value of K in the range from 1 to 20.
- The accuracy for each channel and each K is stored in the `accuracy` list.

The final result is a list of accuracies for each channel and K, allowing us to evaluate the classifier's performance.

2.9.3 Outcome

The best channel and K value are determined based on the highest classification accuracy obtained through cross-validation.

—

Method 2: Frequency-domain Feature Extraction

In this method, we extract frequency-domain features by applying the Fourier Transform to the raw EMG signals. This approach focuses on the frequency components of the signal, which might contain more distinguishable information for classification.

“‘latex

2.10 Method 2: Frequency-domain Feature Extraction

This method involves transforming the raw EMG signal into the frequency domain using the Fourier Transform. By extracting the magnitude of the frequency components, we can analyze the signal's frequency characteristics, which may provide more meaningful features for classification.

2.10.1 Procedure

- First, the raw EMG signal is transformed into the frequency domain using the Fourier Transform.
- The magnitude of the frequency components is extracted.
- The signal is truncated to half its spectrum (since it is symmetric for real signals).
- The extracted frequency-domain features are then used for classification with the KNN classifier.

- Like in Method 1, Leave-One-Out Cross-Validation (LOO-CV) is used to evaluate the performance for different values of K.

2.10.2 Code Explanation

The function `process_frequency_domain_2` handles the frequency-domain feature extraction and classification.

```
def process_frequency_domain_2(raw_trials, labels):
    num_ch = raw_trials[0].shape[1]
    k_values = range(1, 20)
    accuracy = []

    for ch in range(num_ch):
        frequency_features = [Get_freqD_Features(trial[:, ch]) for trial in
            raw_trials]
        accuracies_for_k = [leave_one_out_knn_classification(
            frequency_features, labels, k) for k in k_values]
        accuracy.append(accuracies_for_k)

    return accuracy
```

Listing 2.10: Frequency-domain Feature Extraction

- `raw_trials`: The segmented EMG signal data.
- `labels`: The corresponding labels for each trial.
- For each channel, the function applies the `Get_freqD_Features` function to extract the frequency-domain features.
- The features are then used for classification, and the accuracy is calculated for each K.

2.10.3 Outcome

This method helps identify which frequency components of the signal are most useful for distinguishing between different movement types.

—

Method 3: Combined Features (Time + Frequency)

In this method, we combine the time-domain and frequency-domain features into a single feature vector. The goal is to leverage both types of information to improve classification performance.

“‘latex

2.11 Method 3: Combined Features (Time + Frequency)

This method combines both time-domain and frequency-domain features for each trial. By combining the features from both domains, we aim to use the complementary information from both time and frequency to improve classification accuracy.

2.11.1 Procedure

- Time-domain and frequency-domain features are extracted for each trial.
- The features are concatenated to create a combined feature vector for each trial.
- These combined features are then used for classification.
- The classifier performance is evaluated using Leave-One-Out Cross-Validation (LOO-CV) for different values of K.

2.11.2 Code Explanation

The function `execute_combined_feature_method3` combines the time-domain and frequency-domain features and performs classification.

```
def execute_combined_feature_method3(trials, labels):
    print("\n=== Method 3: Combined Features (Time + Frequency) ===")
    combined_features = [np.concatenate([
        Get_timeD_Features(trial), Get_freqD_Features(trial)
    ]) for trial in trials]

    Kscores = leave_one_out_knn_classification_method3(combined_features,
       labels)
    optimal_k, best_accuracy = find_optimal_k_and_accuracy(Kscores)

    plot_k_vs_accuracy(Kscores, "Combined Features")

    return optimal_k, best_accuracy
```

Listing 2.11: Combined Features (Time + Frequency)

- `trials`: The segmented EMG signal data for each trial.
- `labels`: The labels corresponding to each trial.
- For each trial, the time-domain and frequency-domain features are extracted using `Get_timeD_Features` and `Get_freqD_Features`.
- The features are concatenated to form a combined feature vector for each trial.
- The combined features are then classified using KNN, and the optimal K value is determined by evaluating the classification accuracy across different values of K.

2.11.3 Outcome

Combining both time-domain and frequency-domain features is expected to improve classification performance by using more comprehensive information from the EMG signal.

—

Comparison of Methods

- **Method 1 (Time-domain)**: Focuses on the raw signal, providing simple yet effective features. However, it may not capture all the useful information in the frequency components.

- **Method 2 (Frequency-domain)**: Focuses on the frequency components of the signal, which might capture more distinguishable information for classification. It requires a transformation (Fourier Transform) and might lose some time-domain details.
- **Method 3 (Combined Features)**: Combines the strengths of both time and frequency-domain features. It uses complementary information, potentially improving classification accuracy, but at the cost of higher computational complexity.

Each method has its strengths, and the optimal method depends on the specific application and data characteristics. The combination of time and frequency features often results in the best classification accuracy, as it utilizes both time-domain and frequency-domain information.

2.12 Main Execution Block

The main execution block runs the entire processing pipeline:

- Loads the EMG data and stimulus labels.
- Applies the high-pass filter to the data.
- Extracts time-domain and frequency-domain features.
- Performs classification with KNN and optimizes the K value.
- Plots the accuracy vs K for different feature extraction methods.

Chapter 3

Applying the Filter

The following figures show the **time-domain** spectra for the three channels:

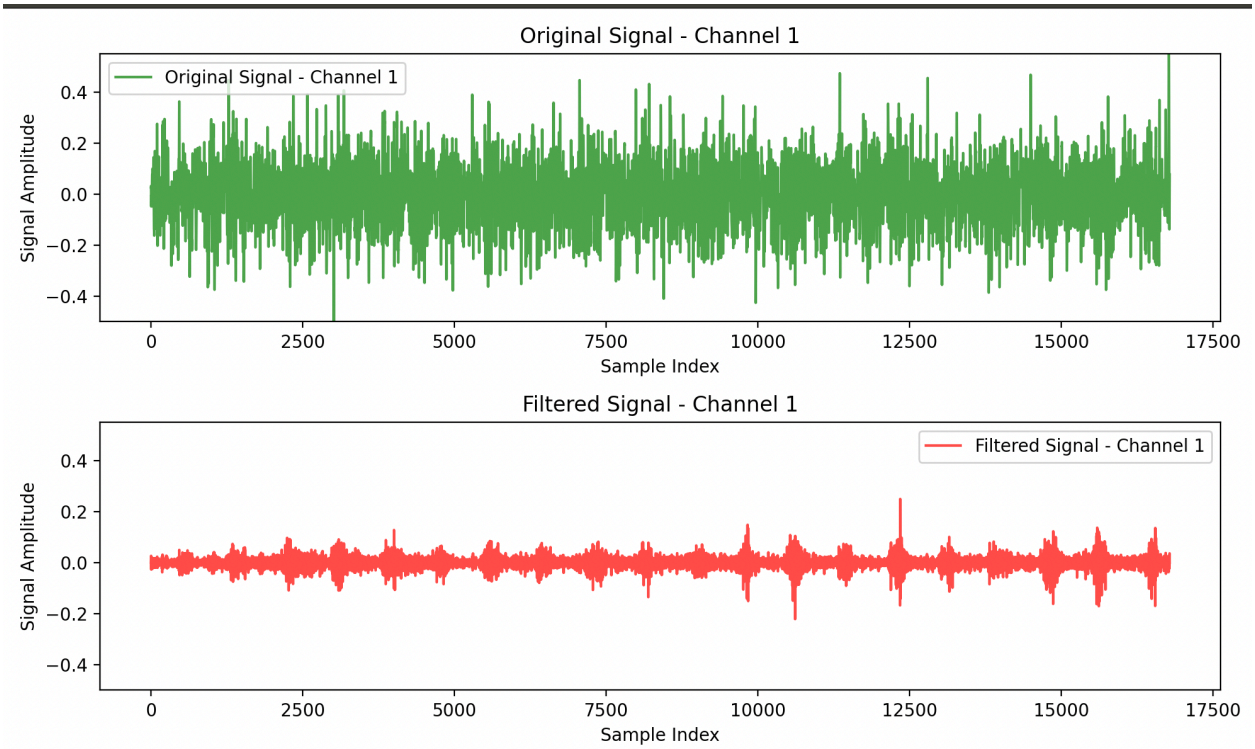

Figure 3.1: Channel 1

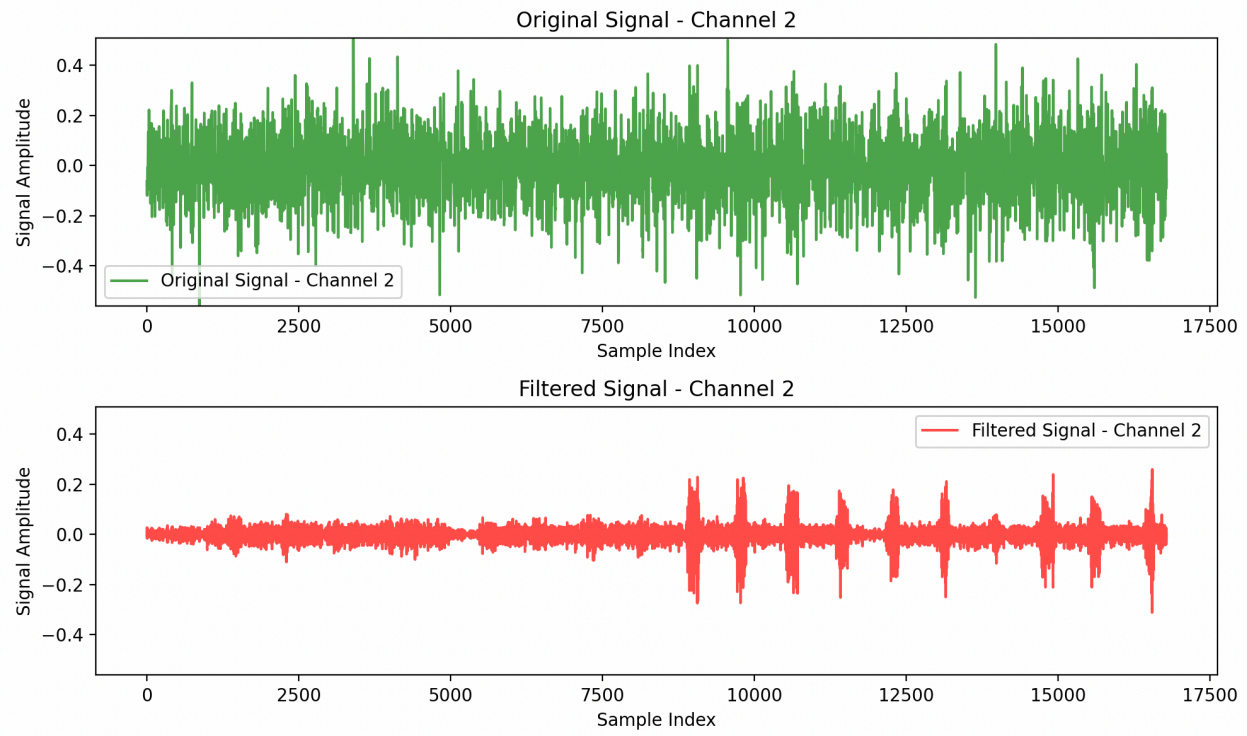

Figure 3.2: Channel 2

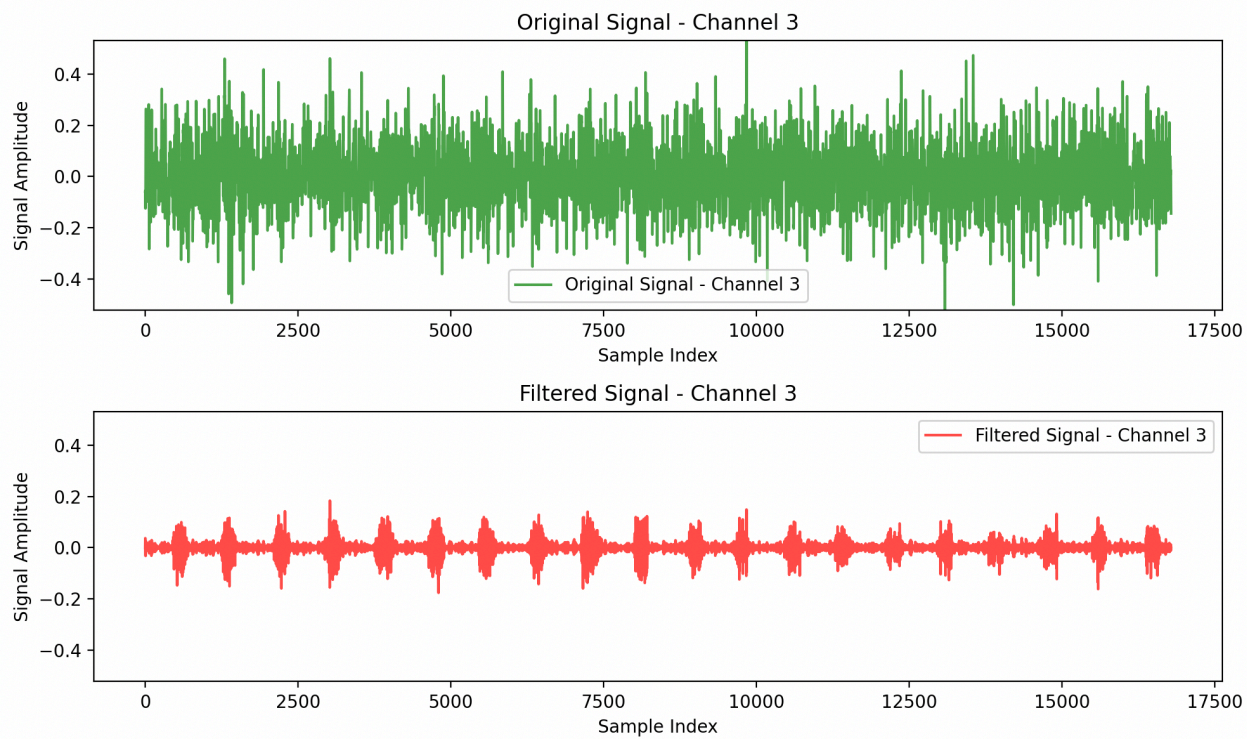

Figure 3.3: Channel 3

The following figures show the **time-domain** spectra before and after filtering for each channel:

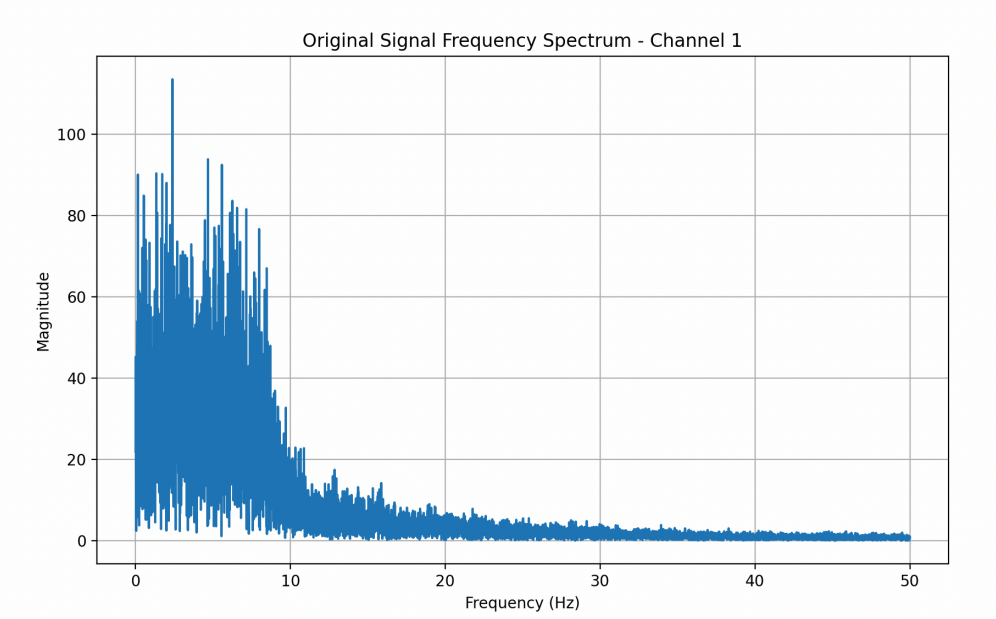

Figure 3.4: Channel 1 before Filtration

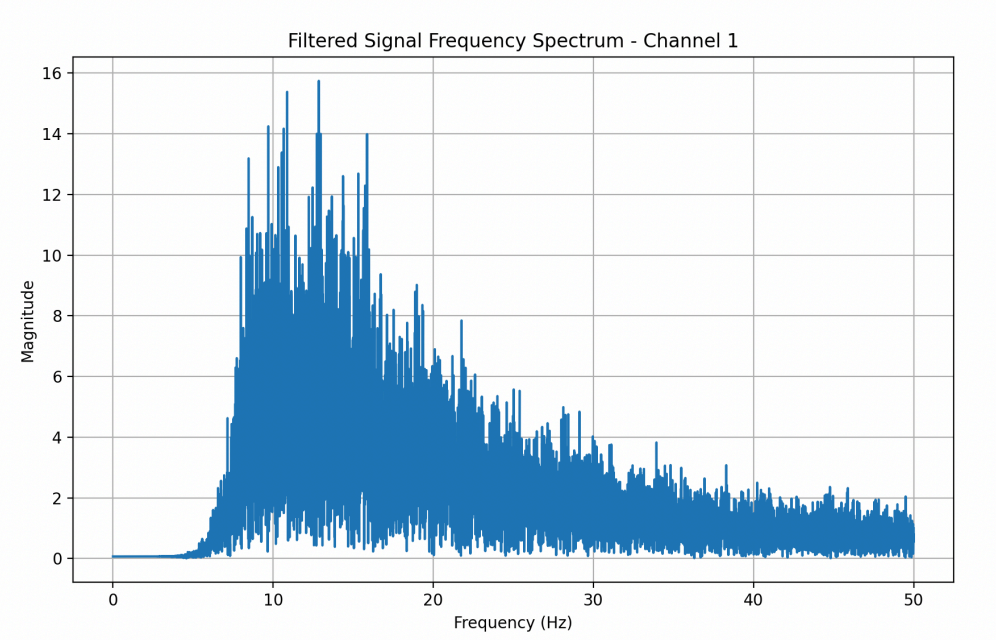

Figure 3.5: Channel 1 after Filtration

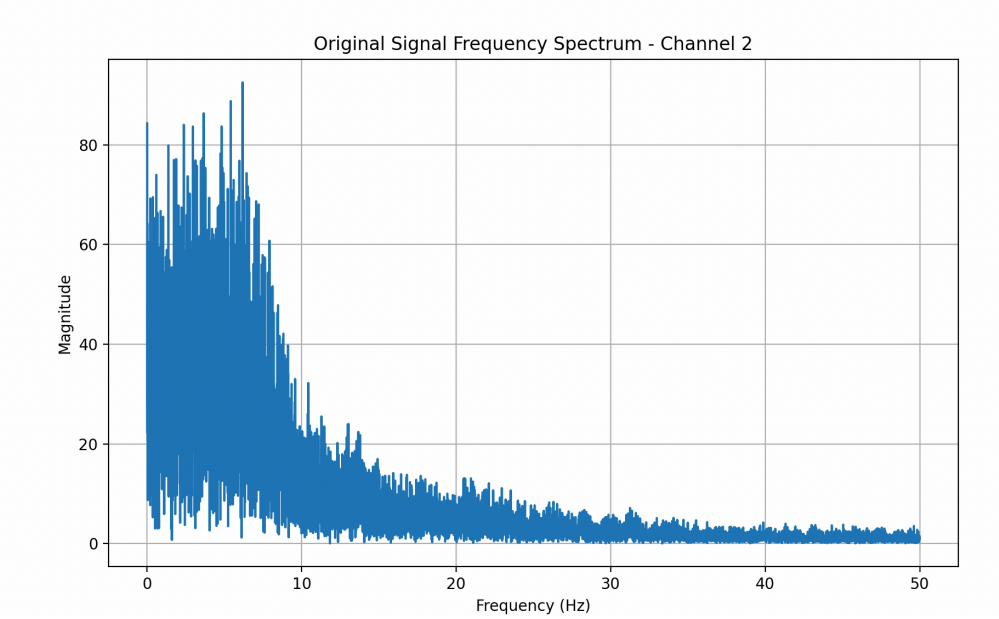

Figure 3.6: Channel 2 before Filtration

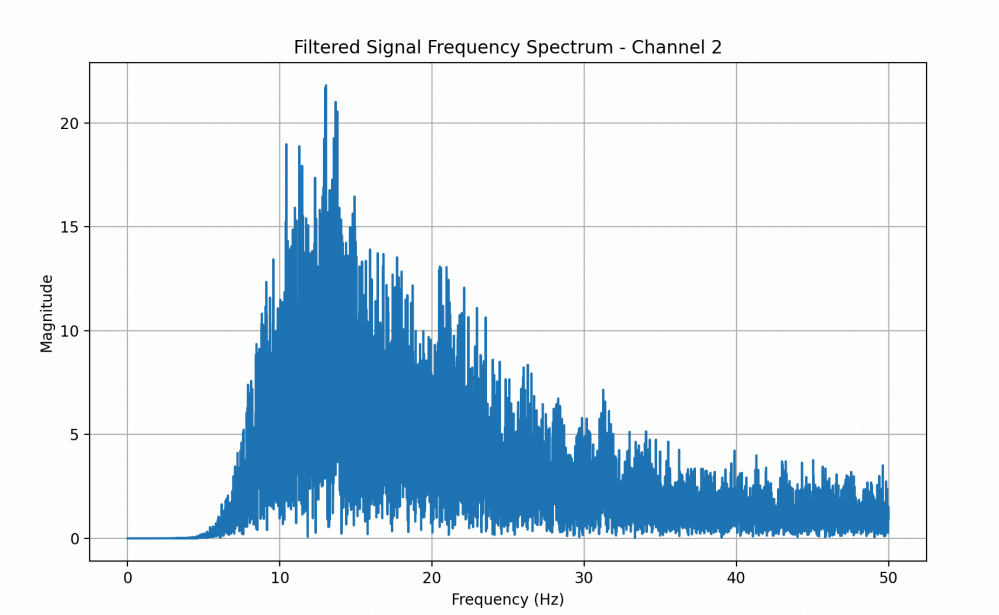

Figure 3.7: Channel 2 after Filtration

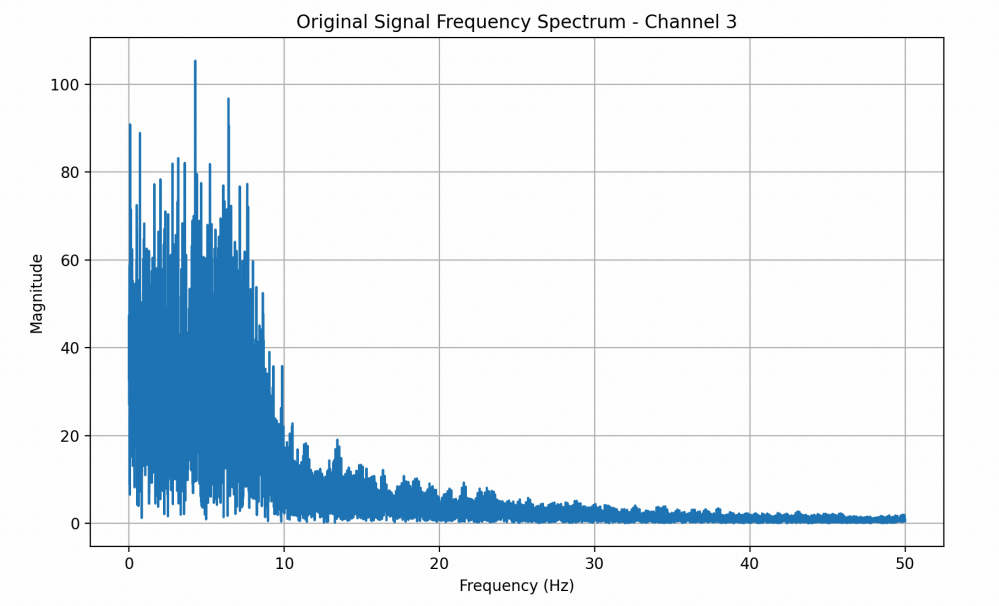

Figure 3.8: Channel 3 before Filtration

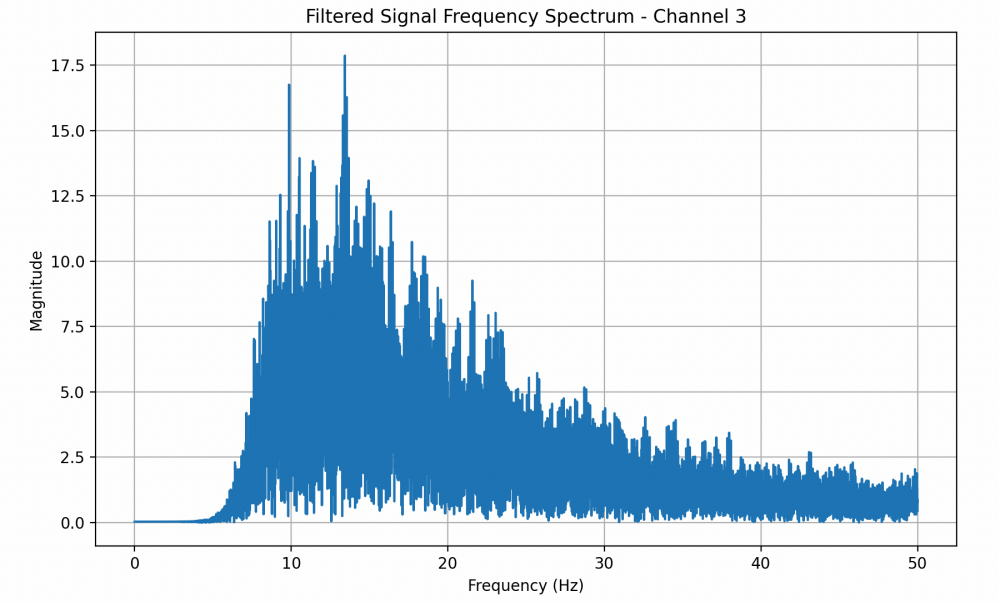

Figure 3.9: Channel 3 after Filtration

3.1 Comments

The time-domain graphs show that the filtered signal is much cleaner, with less low-frequency noise. The signal no longer exhibits slow, fluctuating components, meaning the filter worked well to remove the low-frequency noise while preserving the higher frequencies. In the frequency-domain graphs, we can see that the original signal had significant energy in the

low frequencies, but the filtered signal shows that most of these low frequencies have been reduced, confirming that the filter effectively removed them.

Chapter 4

Highest leave-one-out classification accuracy

4.1 TIME DOMAIN:

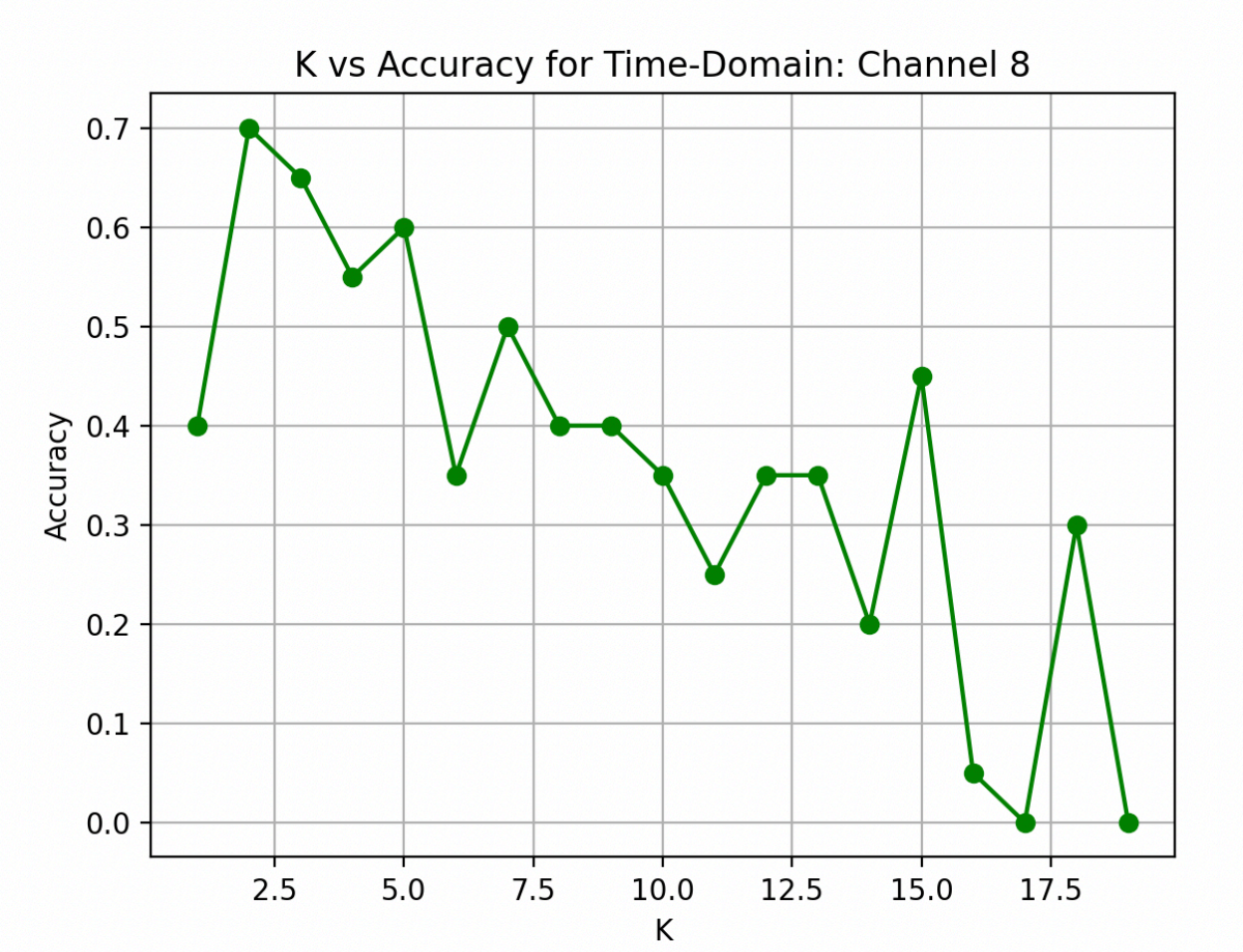

Figure 4.1: Subject 1

- **Subject Number:** 1
- **Channel:** 8
- k**-value:** 2
- **Accuracy:** 70%

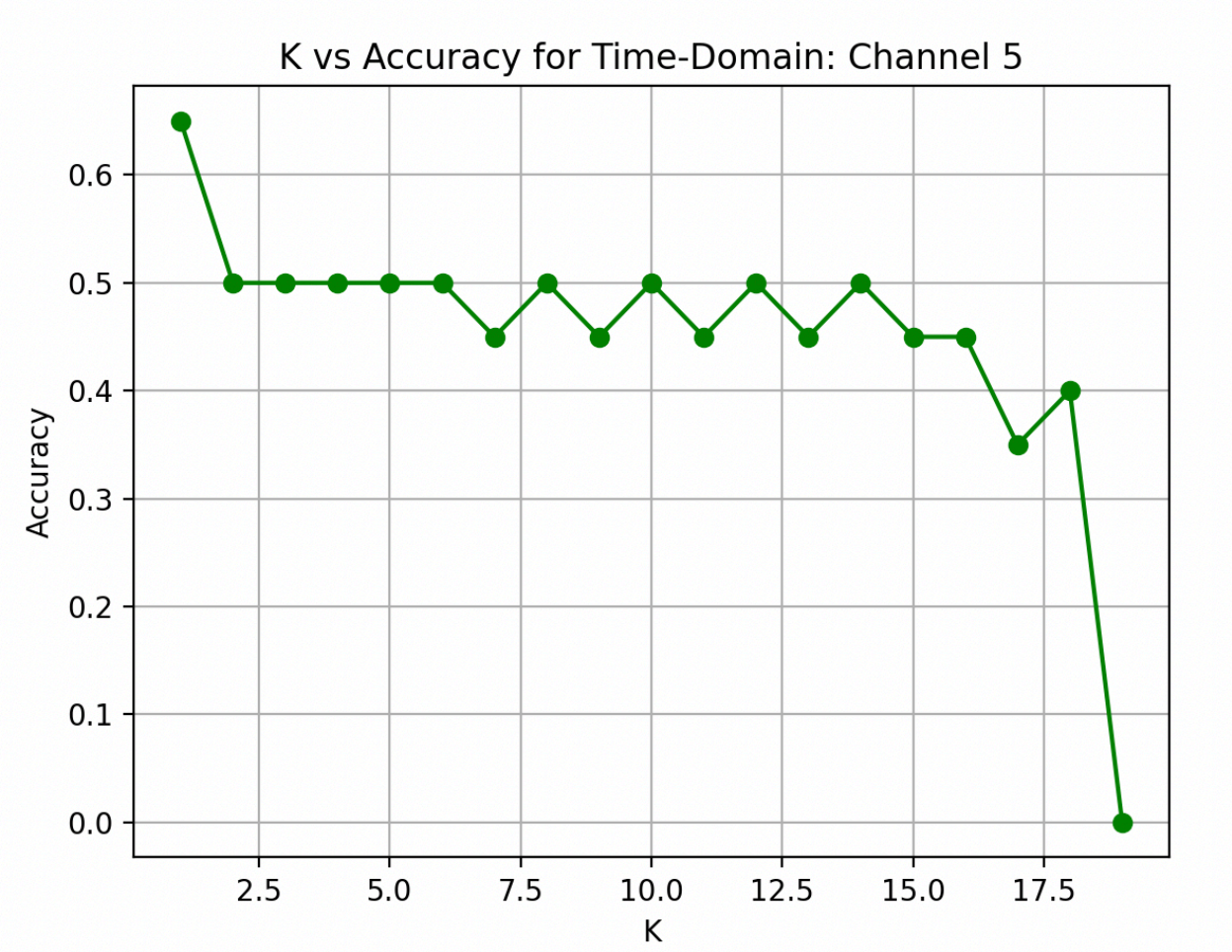

Figure 4.2: Subject 2

- **Subject Number:** 2
- **Channel:** 5
- k**-value:** 1
- **Accuracy:** 65%

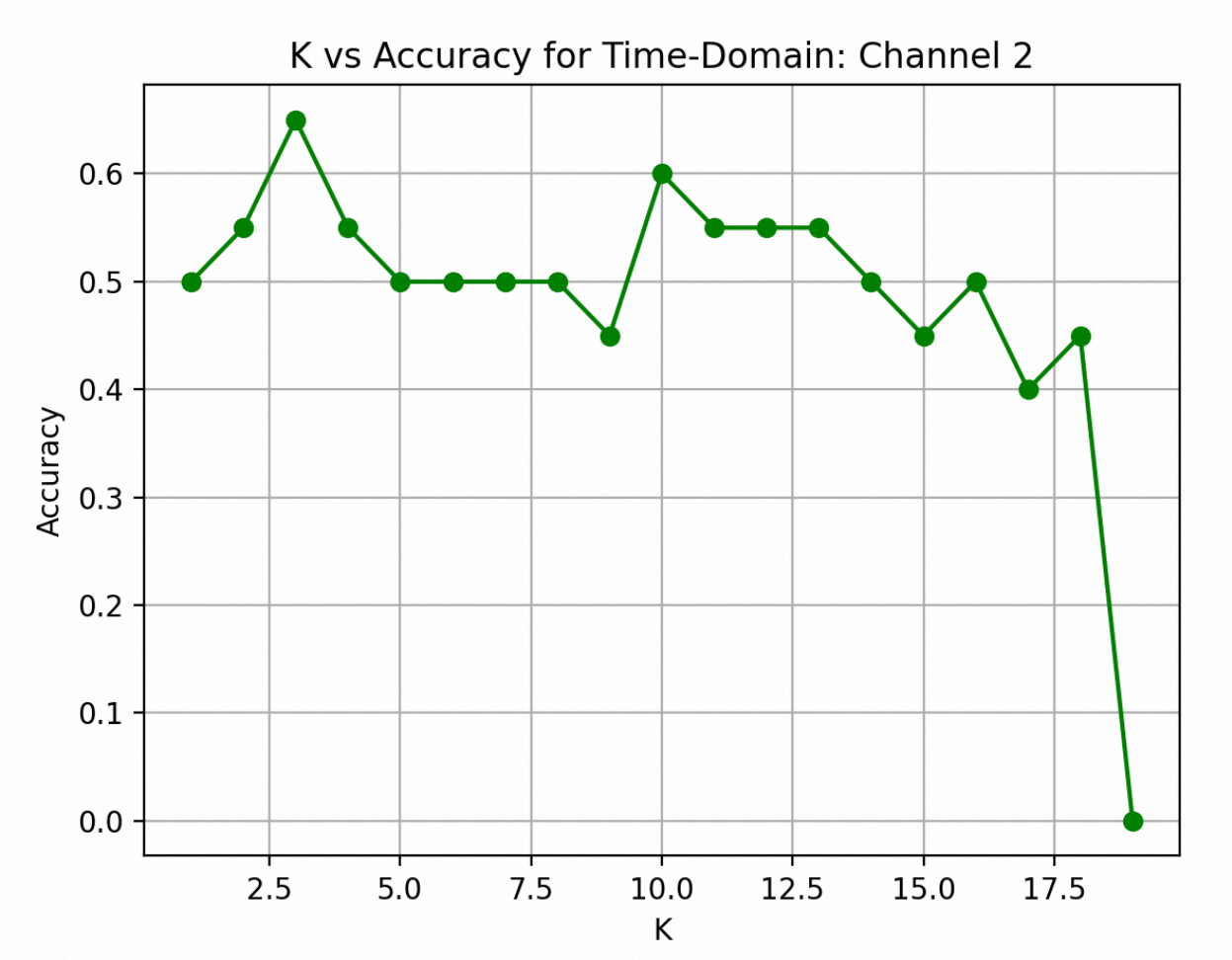

Figure 4.3: Subject 3

- **Subject Number:** 3
- **Channel:** 2
- **k-value:** 3
- **Accuracy:** 65%

4.2 FREQUENCY DOMAIN

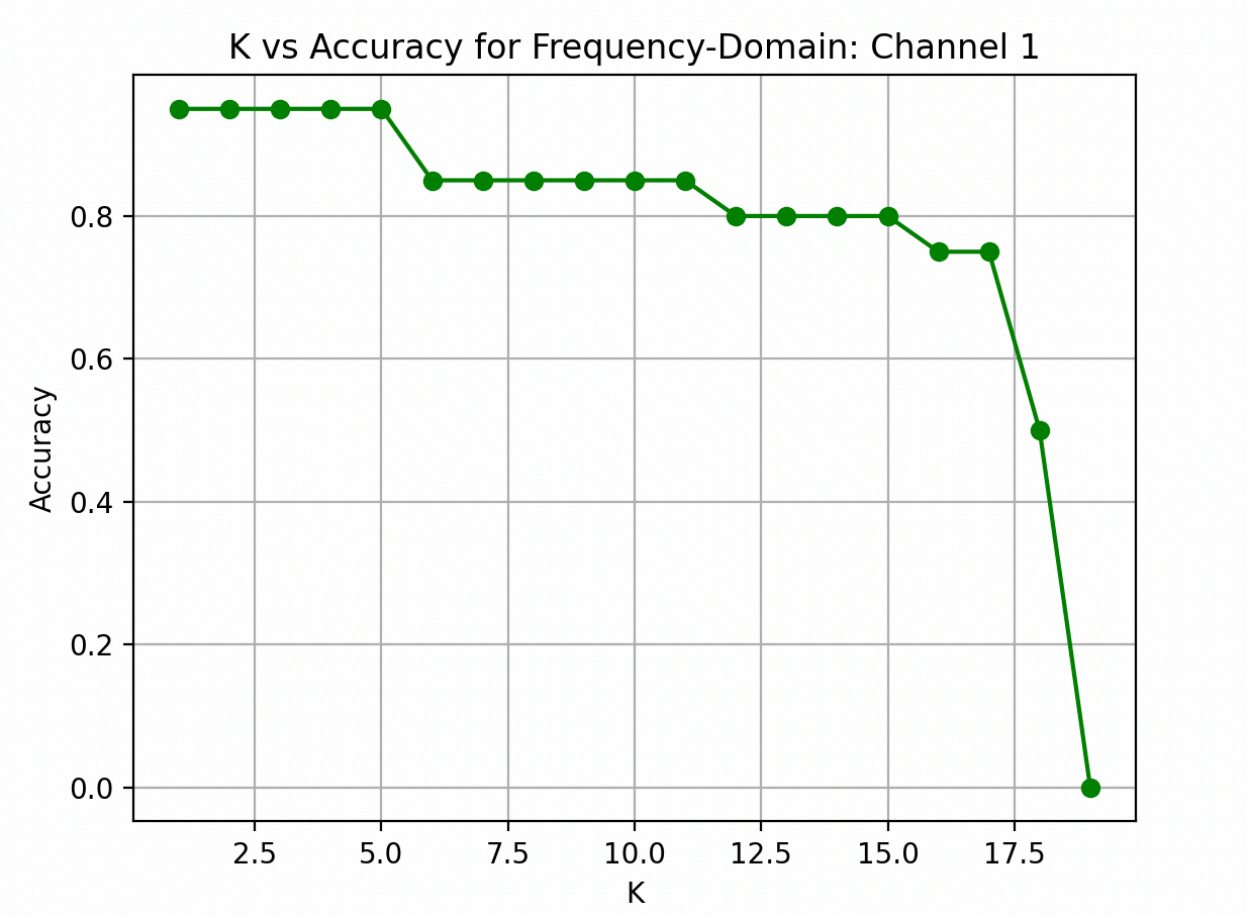

Figure 4.4: Subject 1

- **Subject Number:** 1
- **Channel:** 1
- **k-value:** 1
- **Accuracy:** 95%

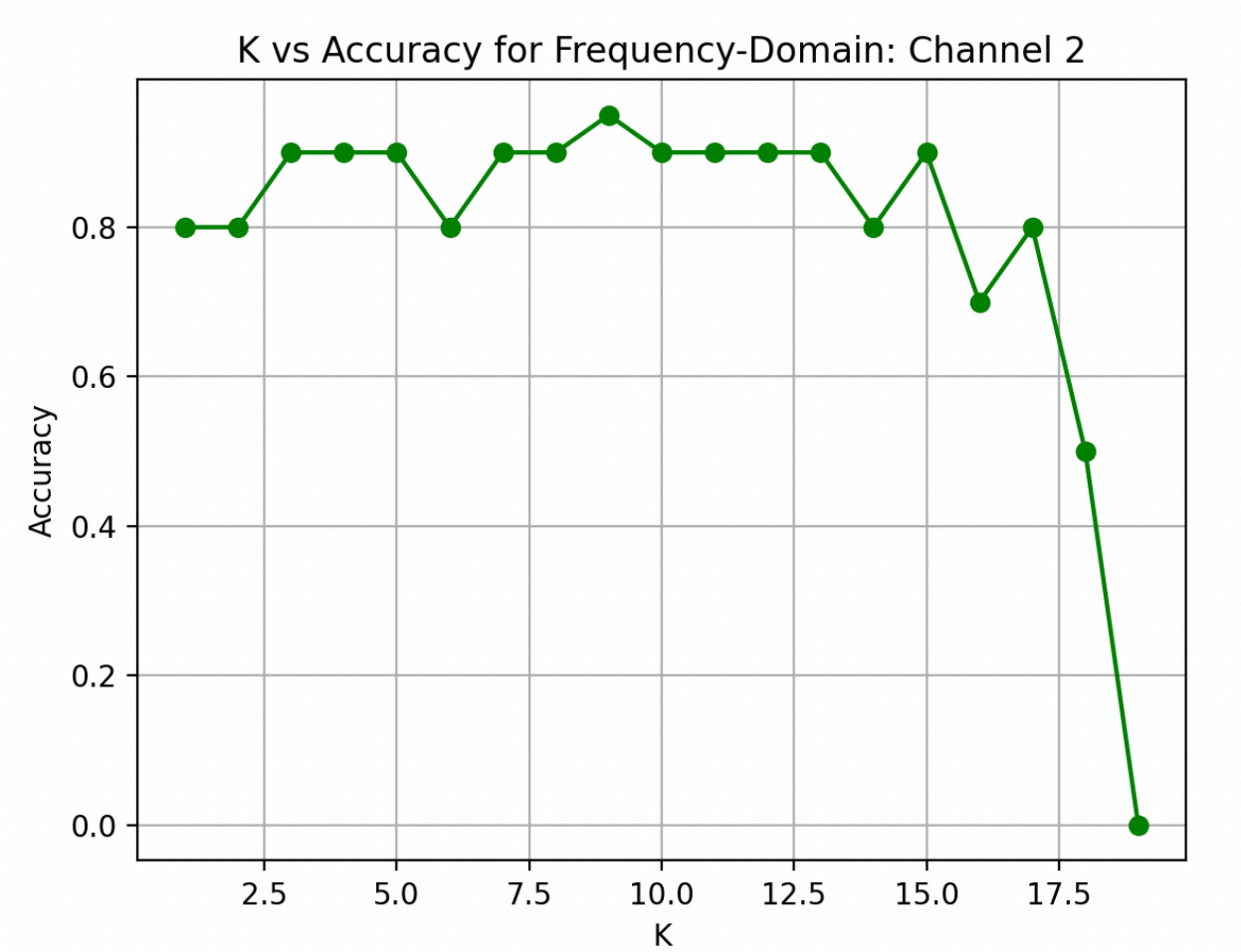

Figure 4.5: Subject 2

- **Subject Number:** 2
- **Channel:** 2
- **k-value:** 9
- **Accuracy:** 95%

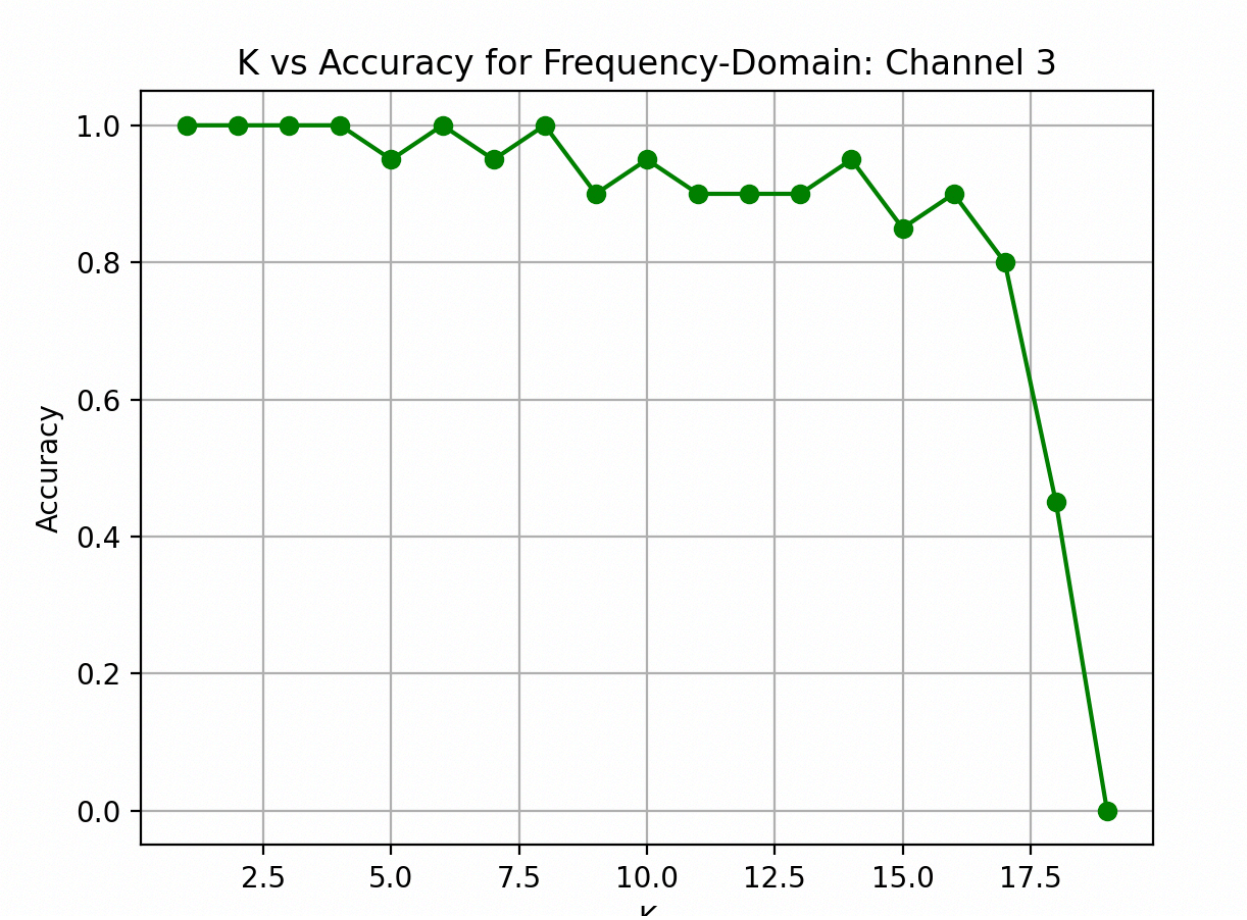

Figure 4.6: Subject 3

- **Subject Number:** 3
- **Channel:** 3
- k**-value:** 1
- **Accuracy:** 100%

4.3 Comments

Across the three subjects, the frequency-domain analysis consistently yields better classification accuracy compared to the time-domain analysis. This suggests that frequency-domain features, such as spectral power and frequency characteristics, are more effective at distinguishing different muscle activities in the sEMG signals. The frequency domain captures subtle differences in muscle movement patterns that may be difficult to discern in the time domain. While time-domain features can provide useful information about signal strength or variability, they may not capture the distinct frequency components that are often crucial for the accurate classification of muscle movements.

It was also observed that lower values of k in the KNN classifier tended to give better performance than higher values for the same channel. This could be due to the classifier's

increased sensitivity to smaller, local variations in the data when k is low, which may help distinguish finer differences in muscle activation.

Regarding the different channels, no consistent trend emerged that indicated better performance from any specific channel. This could be attributed to the fact that each channel captures a unique aspect of the muscle signals. The features most effective for classification may vary depending on the muscle being targeted or the specific type of movement. Therefore, combining features from both the time domain and frequency domain across multiple channels could offer a more complete and robust representation of muscle activity, potentially leading to improved classification performance.

Chapter 5

Combining Data from All Channels

5.1 Combining Data from Different Channels

To combine data from all channels, we first extracted features individually for each channel, ensuring that the distinct characteristics of each channel's signal were maintained. This approach helps capture the spatial and temporal variations inherent in multi-channel sEMG data. After extracting these features, they were merged into a single feature vector. This process allows us to take advantage of the unique contribution of each channel while forming a consolidated feature set for further processing.

We explored three different strategies for concatenating the data:

1. **Time-domain Features:** In this approach, the features extracted from the time-domain signals of all channels are combined into a single vector. This method maintains the temporal characteristics of the muscle activity.
2. **Frequency-domain Features:** Here, the frequency-domain features from all channels are concatenated. By focusing on the frequency components of the signal, this method captures different muscle movement patterns, which can be crucial for classification.
3. **Combining Time-domain and Frequency-domain Features:** This strategy involves concatenating both time-domain and frequency-domain features from each channel into one large feature vector. By merging both types of information, this method aims to provide a more comprehensive view of the muscle activity.

Each approach involves stacking the features from all channels into one unified vector, which is then used as input for classification.

5.2 Effect of Including All Channels on Classification Performance

Using data from all channels can significantly influence the performance of the classification system. Below is an analysis of the effects of each concatenation approach:

- **Time-domain Features:** When we combined the time-domain features from all channels, the classification accuracy decreased. This suggests that including raw time-

domain signals from multiple channels may introduce noise or redundant data, which hampers the classifier's ability to distinguish patterns effectively.

- **Frequency-domain Features:** Combining frequency-domain features from all channels yielded classification accuracy close to that of individual channels. This suggests that frequency-domain features are more consistent and robust, capturing essential muscle activity patterns that contribute effectively to classification.
- **Time-domain and Frequency-domain Features Combined:** For the combined features approach, the accuracy was found to fluctuate between the results obtained from the time-domain and frequency-domain methods. This could imply that the combined features introduce conflicting information, leading to variable results.

These results indicate that while merging data from all channels can improve the classifier's performance, the choice between time-domain and frequency-domain features plays a crucial role. Combining both features does not always lead to better performance and may require further fine-tuning to determine the best combination.

5.3 Performance Comparison of the Three Approaches

The following plot shows a comparison of the three feature concatenation approaches. It visualizes how the classification accuracy differs when using time-domain, frequency-domain, and combined features.

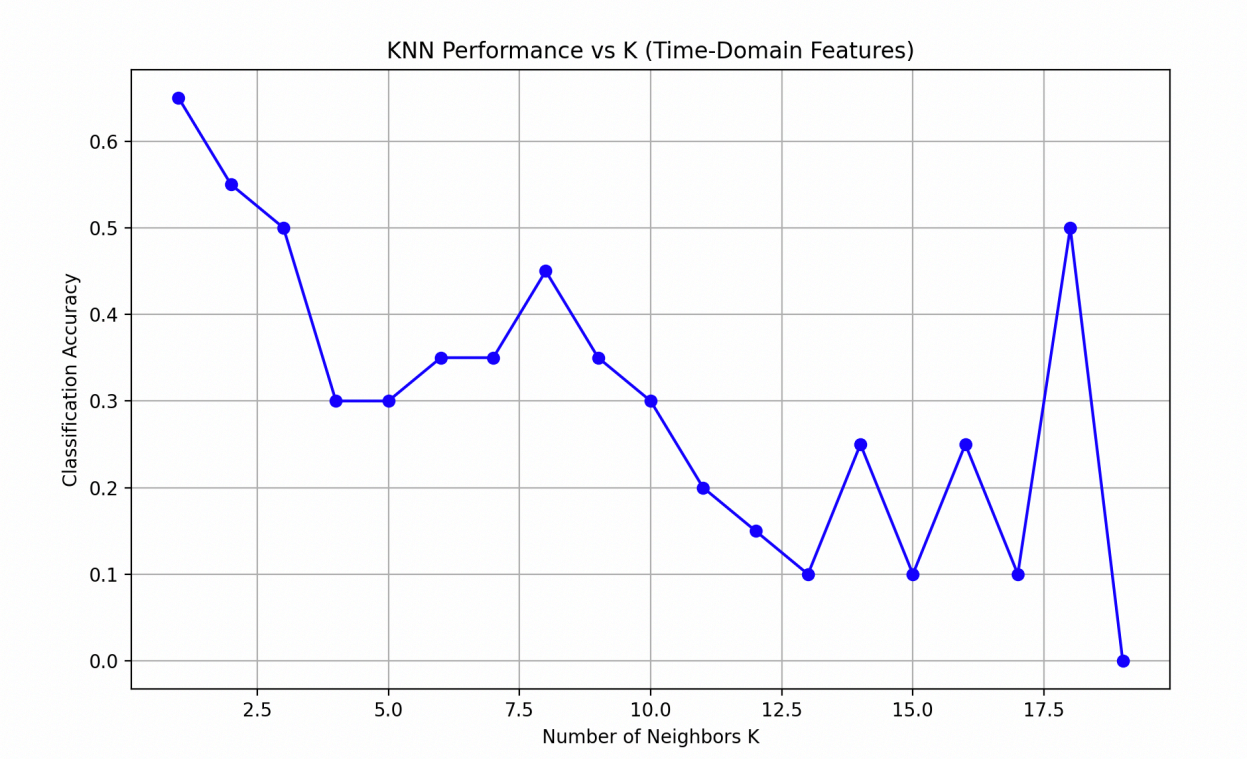

Figure 5.1: Time Domain Combined

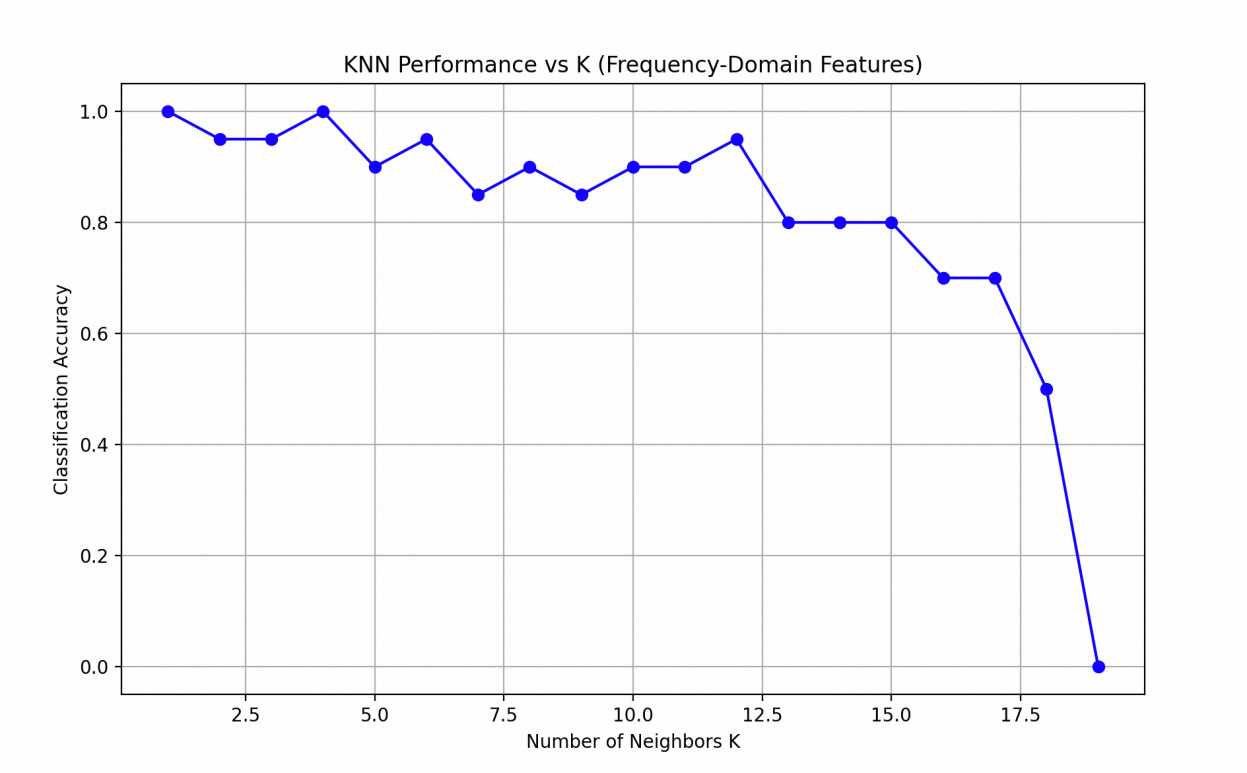

Figure 5.2: Frequency Domain Combined

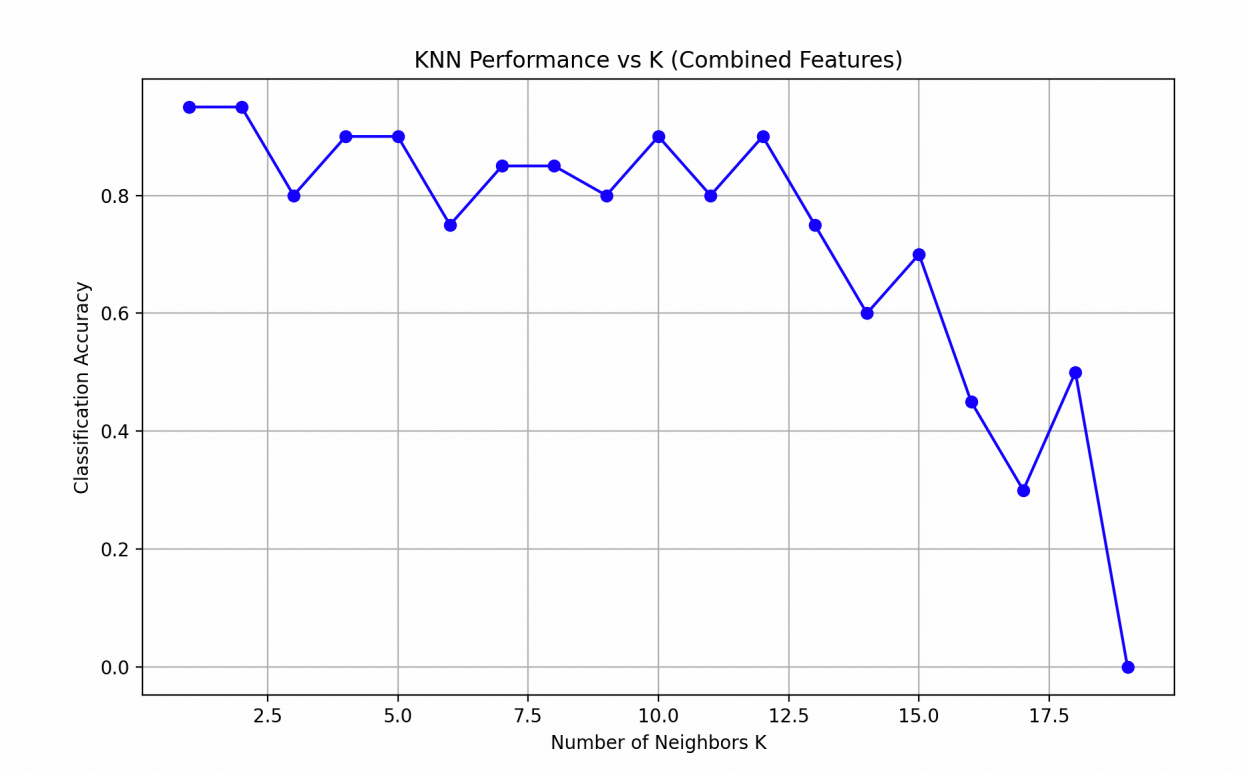

Figure 5.3: All Data Combined