DSP Project Report

Report Group Members:

Ismaiel Sabet 900221277 ismaielsabet@aucegypt.com

Mohamed El Refai 900222334 mohamedref11@aucegypt.edu

Content

Introduction

In this project, we set out to assess the attention state of subjects using provided EEG waves. To achieve this goal, we built the project in python, trying our best to maintain modularity and cohesion.

Approach used

Firstly, let’s attach pictures of the actual code. Following that, an explanation of the approach will be provided.

```

# Ismaiel Sabet 900221277
# Mohamed El-Refai 900222334
# DSP project


import numpy as np
from scipy.io import loadmat
from sklearn.neighbors import KNeighborsClassifier
import pandas as pd
from numpy.fft import rfft, rfftfreq

#Algorithm

    # Step 1: Compute the CAR filter to the data
    # Step 2:
        #   2.1: For each electrode of the *7* electrodes,
        #   2.2: For each trial of each class of attention (focused versus drowsy), compute the Fourier Transform and compute the power in the delta, theta,
        #   2.3: Apply KNN
        #   2.4: Compute the classification error for each value of K (from 2.3)

bands = {
    'Delta': (0.5, 4),
    'Theta': (4, 8),
    'Alpha': (8, 13),
    'Beta': (13, 30),
    'Gamma': (30, 45)
}
```

```python
# For consistent ordering of bands in feature arrays
band_names_ordered = list(bands.keys())

def apply_car(data):
    return data - np.mean(data, axis=2, keepdims=True) #because data has dimension of 3 and the count starts from 0

def compute_band_power(signal, fs, band_definitions):
    # calculates power in specified frequency bands for a single signal trace
    # signal: 1D numpy array (time-series data for one trial, one channel)
    # fs: sampling frequency
    # band_definitions: dictionary defining band names and their (low_freq, high_freq)

    n = len(signal)
    if n == 0: # Handle empty signal case
        return {band_name: 0 for band_name in band_definitions}

    yf = rfft(signal)  # Compute the FFT
    power_spectrum = np.abs(yf)**2  # Compute power (magnitude squared)
    xf = rfftfreq(n, 1/fs)  # Get the frequencies

    # Calculate the power in each band
    band_powers = {}
    for band_name, (low_freq, high_freq) in band_definitions.items():
        band_mask = (xf >= low_freq) & (xf < high_freq) # Frequencies within the band

        # Calculate mean power in the band (if there are frequencies in the band)
        if np.any(band_mask) and len(power_spectrum[band_mask]) > 0:
            band_powers[band_name] = np.mean(power_spectrum[band_mask])
        else:
            band_powers[band_name] = 0

    return band_powers

```

```python
def extract_features(data_car, fs, band_definitions, ordered_band_names_list):
    # Extracts band power features for all trials and channels and bands
    # data_car: CAR filtered data (trials, samples, channels)
    # fs: sampling rate
    # band_definitions: dictionary of band names and (low, high) frequencies
    # ordered_band_names_list: list of band names to ensure consistent feature order
    # Returns: 3D numpy array (trials, channels, bands_ordered)

    n_trials, _, n_channels = data_car.shape
    n_bands = len(ordered_band_names_list)

    features_3d = np.zeros((n_trials, n_channels, n_bands))

    for t in range(n_trials):
        for ch in range(n_channels):
            signal_one_trial_one_channel = data_car[t, :, ch]

            # Compute band powers for this specific signal
            powers = compute_band_power(signal_one_trial_one_channel, fs, band_definitions)

            # Store them in the 3D feature array in the specified order
            for b_idx, b_name in enumerate(ordered_band_names_list):
                features_3d[t, ch, b_idx] = powers.get(b_name, 0) # Default to 0 if band missing

    return features_3d

#main logic
results_list = []
data_base_path = '/Users/refobic/Downloads/Project/Data/'

```

```python
for subj in range(1, 6):
    print(f"\nProcessing Subject {subj}...")

    try:     #load training and testing data
        train_mat = loadmat(f'{data_base_path}train_data_{subj}.mat')
        test_mat = loadmat(f'{data_base_path}test_data_{subj}.mat')
    except FileNotFoundError:
        print(f"Error: Data files for subject {subj} not found in {data_base_path}. Skipping.")
        continue

    data_train = train_mat['data']  # trials x samples x channels
    labels_train = train_mat['labels'].squeeze()
    data_test = test_mat['data']
    labels_test = test_mat['labels'].squeeze()

    fs = float(train_mat['fs'].squeeze())
    # Handle different structures of 'channels' in .mat files
    if train_mat['channels'].ndim == 1 or train_mat['channels'].shape[0] == 1 or train_mat['channels'].shape[1] == 1:
        channels = [str(c[0]) if isinstance(c, np.ndarray) and c.size > 0 and isinstance(c[0], (np.str_, str))
                    else str(c) if isinstance(c, (np.str_, str))
                    else f"Ch{i+1}" # Fallback name
                    for i, c in enumerate(train_mat['channels'].squeeze())]
    elif train_mat['channels'].ndim == 2: # Often (1, num_channels) with cell arrays
         channels = [str(train_mat['channels'][0, i][0]) if isinstance(train_mat['channels'][0, i], np.ndarray) and train_mat['channels'][0,i].size > 0
                    else str(train_mat['channels'][0, i])
                    for i in range(train_mat['channels'].shape[1])]
    else: # Fallback if unsure
        channels = [f"Ch{i+1}" for i in range(data_train.shape[2])]

    n_channels = data_train.shape[2]
    n_bands_defined = len(band_names_ordered)

```

```python
    # Step 1: Apply Common Average Reference (CAR) filtering
    data_train_car = apply_car(data_train)
    data_test_car = apply_car(data_test)

    # Step 2.1 & 2.2: Extract band power features for all channels and bands
    # This creates a 3D matrix: (trials, channels, bands)
    features_train_3d = extract_features(data_train_car, fs, bands, band_names_ordered)
    features_test_3d = extract_features(data_test_car, fs, bands, band_names_ordered)

    # Step 2.3 & 2.4

    # Scenario 1: Single Channel & Single Band
    print("  Scenario 1: Single Channel & Band")
    for ch_idx, ch_name in enumerate(channels):
        for b_idx, b_name in enumerate(band_names_ordered):
            X_tr = features_train_3d[:, ch_idx, b_idx].reshape(-1, 1)
            X_te = features_test_3d[:, ch_idx, b_idx].reshape(-1, 1)

            best_acc_s1, best_k_s1 = 0.0, 1
            for K_val in range(1, 11):
                if K_val > len(X_tr): break #n_neighbors must be less than n_samples
                knn = KNeighborsClassifier(n_neighbors=K_val)
                knn.fit(X_tr, labels_train)
                acc = knn.score(X_te, labels_test)
                if acc > best_acc_s1:
                    best_acc_s1, best_k_s1 = acc, K_val
            results_list.append({
                'Subject': subj, 'Scenario': 'Single Channel & Band',
                'Channel': ch_name, 'Band': b_name,
                'Best K': best_k_s1, 'Accuracy': best_acc_s1
            })

```

```python
        # Scenario 2: Single Channel & All Bands
        print("  Scenario 2: Single Channel, All Bands")
        for ch_idx, ch_name in enumerate(channels):
            X_tr = features_train_3d[:, ch_idx, :] # Features: (trials, all_bands_for_this_channel)
            X_te = features_test_3d[:, ch_idx, :]

            best_acc_s2, best_k_s2 = 0.0, 1
            for K_val in range(1, 11):
                if K_val > len(X_tr): break
                knn = KNeighborsClassifier(n_neighbors=K_val)
                knn.fit(X_tr, labels_train)
                acc = knn.score(X_te, labels_test)
                if acc > best_acc_s2:
                    best_acc_s2, best_k_s2 = acc, K_val
            results_list.append({
                'Subject': subj, 'Scenario': 'Single Channel, All Bands',
                'Channel': ch_name, 'Band': 'All',
                'Best K': best_k_s2, 'Accuracy': best_acc_s2
            })

        # Scenario 3: Single Band & All Channels
        print("  Scenario 3: Single Band, All Channels")
        for b_idx, b_name in enumerate(band_names_ordered):
            X_tr = features_train_3d[:, :, b_idx] # Features: (trials, all_channels_for_this_band)
            X_te = features_test_3d[:, :, b_idx]

            best_acc_s3, best_k_s3 = 0.0, 1
            for K_val in range(1, 11):
                if K_val > len(X_tr): break
                knn = KNeighborsClassifier(n_neighbors=K_val)
                knn.fit(X_tr, labels_train)
                acc = knn.score(X_te, labels_test)
                if acc > best_acc_s3:
                    best_acc_s3, best_k_s3 = acc, K_val
            results_list.append({
                'Subject': subj, 'Scenario': 'Single Band, All Channels',
                'Channel': 'All', 'Band': b_name,
                'Best K': best_k_s3, 'Accuracy': best_acc_s3
            })

        # Scenario 4: All Channels & All Bands
        print("  Scenario 4: All Channels & All Bands")
        X_tr = features_train_3d.reshape(features_train_3d.shape[0], n_channels * n_bands_defined)
        X_te = features_test_3d.reshape(features_test_3d.shape[0], n_channels * n_bands_defined)

        best_acc_s4, best_k_s4 = 0.0, 1
        for K_val in range(1, 11):
            if K_val > len(X_tr): break
            knn = KNeighborsClassifier(n_neighbors=K_val)
            knn.fit(X_tr, labels_train)
            acc = knn.score(X_te, labels_test)
            if acc > best_acc_s4:
                best_acc_s4, best_k_s4 = acc, K_val
        results_list.append({
            'Subject': subj, 'Scenario': 'All Channels & All Bands',
            'Channel': 'All', 'Band': 'All',
            'Best K': best_k_s4, 'Accuracy': best_acc_s4
        })
```

```python
# output
df_results = pd.DataFrame(results_list)
output_csv_path = f'{data_base_path}../EEG_KNN_Results_Detailed.csv' # Save one level up from Data folder
df_results.to_csv(output_csv_path, index=False)
print(f"\nDetailed results saved to {output_csv_path}")

# For Deliverable 2 Identify the frequency band and channel and value of K for KNN that gets the highest accuracy on test data for each subject.
best_single_combo_results_list = []
for subj_num in range(1, 6):
    subject_df = df_results[
        (df_results['Subject'] == subj_num) &
        (df_results['Scenario'] == 'Single Channel & Band')
    ]
    if not subject_df.empty:
        best_row = subject_df.loc[subject_df['Accuracy'].idxmax()]
        best_single_combo_results_list.append({
            'Subject': subj_num,
            'Best Channel': best_row['Channel'],
            'Best Band': best_row['Band'],
            'Best K for Combo': best_row['Best K'],
            'Highest Accuracy (Single Combo)': best_row['Accuracy']
        })
    else:
        #this case should not happen if subjects 1-5 are processed and files exist
        best_single_combo_results_list.append({
            'Subject': subj_num, 'Best Channel': 'N/A', 'Best Band': 'N/A',
            'Best K for Combo': 'N/A', 'Highest Accuracy (Single Combo)': 0.0
        })

df_best_single_per_subject = pd.DataFrame(best_single_combo_results_list)
print("\n--- Best Single Channel/Band Performance per Subject (for Deliverable 2) ---")
print(df_best_single_per_subject)
```

```python

best_single_output_path = f'{data_base_path}../EEG_Best_Single_Combo_Per_Subject.csv'
df_best_single_per_subject.to_csv(best_single_output_path, index=False)
print(f"Best single combo per subject results saved to {best_single_output_path}")

print("\nProcessing Complete.")
```

The first thing that we do is include a bunch of imports of functions that will be used in the implementation.

The second thing we do is we define the signal bands. This is based on the segmentation provided in slide 6 of the AttentionState pdf.

Following that, we start dividing the algorithm into different functions. The way we do it is by taking each individual step in the algorithm on slide 10 and making it into a function.

The first of these functions is the apply_car function. This function just applies the CAR filter to our data. We use the methodology expressed in slide 11 and then implement it. Axis has a value of 2 because our data has a dimension count of 3, and the count starts from 0 (so 0, 1, 2).

The second of these functions is the compute_band_power function. This function basically does steps 2.1.1 and 2.1.2 from the algorithm for us, so it computes both the fourier transform and the power in each of the bands. We get the frequencies and the power straightforward by just using the functions. The band power requires some work, and we do that by getting the mean power in those bands, if any such frequencies exist in the band. Even though it's not explicitly mentioned, we do loop over all the electrodes and over the focused and drowsy states, however in regards to the latter, we chose to group them together.

The third of these functions is extract_features. It covers step 2.2 in the algorithm, which deals with creating a feature vector. It first segments the number of trials and the number of channels, then it creates a 3d features vector. Then, the compute_band_power function is called and the powers for the bands are stored in a particular order.

What follows that is the main of the code. We call all the previously declared functions in some form. Then, we reach steps 2.3 and 2.4 of the algorithm. Since we decided to process our KNN across the 4 possibilities of channel and band combinations, we decided not to make it as a function. For the single channel single band case, we extract a single feature per trial, train a k-NN classifier (with k from 1 to 10), and record the best accuracy and corresponding k. For the single channel multiple band case, we use all band powers for the channel as features and we test different values of K until we find the best KNN model. For the single band multiple channels case, we use the band power from all channels for each frequency and evaluate KNN for different k values. Finally for the multiple channel multiple bands case, we use the entire feature set and test all values of K. Note that this code serves as the proposed method for the questions in the report.

The plotting program loads the accuracy results from the .CSV file and puts it into a matrix then reshapes the results into a matrix and produces a grouped bar chart as well as a heatmap which can be seen in the output results later

Output

For each of the 5 subjects, we identified the single channel, frequency band, and K that got us the highest test accuracy which can be seen in the output file "EEG_Best_Single_Combo_Per_Subject.csv"

EEG_Best_Single_Combo_Per_Subject

Subject	Best Channel	Best Band	Best K for Combo	Highest Accuracy (Single Combo)
1	C3	Alpha	7	0.861111111111110
2	Pz	Alpha	3	0.972222222222220
3	C3	Theta	2	0.958333333333330
4	F4	Alpha	5	0.907692307692308
5	C4	Delta	5	0.805555555555560

We can see that **Alpha** was best for subjects 1, 2 and 4 and **Theta** for subject 3 and **Delta** for subject 5. K values ranged from 2 to 7 showing us subject specific smoothness.

Single Channel, all bands

Here we combined all five bands for each channel and re ran KNN and from the result we can see that subject 1, 3 and 5 saw some gains whilst subject 2 and 4 decreased a bit. Which means that for some subjects, all bands can boost performance but it can also introduce noise in some.

Subject	Scenario	Channel	Band	Best K	Accuracy
1	Single Channel, All Bands	F3	All	3	0.5833333333333330
1	Single Channel, All Bands	F4	All	9	0.6111111111111110
1	Single Channel, All Bands	Fz	All	1	0.8055555555555560
1	Single Channel, All Bands	C3	All	1	0.75
1	Single Channel, All Bands	C4	All	1	0.7916666666666670
1	Single Channel, All Bands	Cz	All	7	0.7083333333333330
1	Single Channel, All Bands	Pz	All	3	0.8888888888888890
2	Single Channel, All Bands	F3	All	1	0.6527777777777780
2	Single Channel, All Bands	F4	All	1	0.7222222222222220
2	Single Channel, All Bands	Fz	All	1	0.5694444444444440
2	Single Channel, All Bands	C3	All	1	0.6805555555555560
2	Single Channel, All Bands	C4	All	9	0.8611111111111110
2	Single Channel, All Bands	Cz	All	1	0.5416666666666670
2	Single Channel, All Bands	Pz	All	1	0.9305555555555560
3	Single Channel, All Bands	F3	All	2	0.9027777777777780
3	Single Channel, All Bands	F4	All	5	0.7777777777777780
3	Single Channel, All Bands	Fz	All	3	0.9583333333333330
3	Single Channel, All Bands	C3	All	4	1.0
3	Single Channel, All Bands	C4	All	5	0.9305555555555560
3	Single Channel, All Bands	Cz	All	1	0.7083333333333330
3	Single Channel, All Bands	Pz	All	7	0.9583333333333330
4	Single Channel, All Bands	F3	All	3	0.6923076923076920
4	Single Channel, All Bands	F4	All	3	0.8461538461538460
4	Single Channel, All Bands	Fz	All	6	0.6
4	Single Channel, All Bands	C3	All	5	0.6461538461538460
4	Single Channel, All Bands	C4	All	9	0.8461538461538460
4	Single Channel, All Bands	Cz	All	2	0.5230769230769230
4	Single Channel, All Bands	Pz	All	1	0.7230769230769230
5	Single Channel, All Bands	F3	All	7	0.5833333333333330
5	Single Channel, All Bands	F4	All	10	0.5277777777777780
5	Single Channel, All Bands	Fz	All	10	0.6527777777777780
5	Single Channel, All Bands	C3	All	1	0.7222222222222220
5	Single Channel, All Bands	C4	All	8	0.8888888888888890
5	Single Channel, All Bands	Cz	All	5	0.6111111111111110
5	Single Channel, All Bands	Pz	All	4	0.7361111111111110

All channels, single band

We combined all seven channels for each band and we can see that the result is generally lower as subjects 1, 3, 4 and 5 decreased while subject 2 remained the same. This shows that combining channels does not uniformly capture the most discriminative signal for attention state.

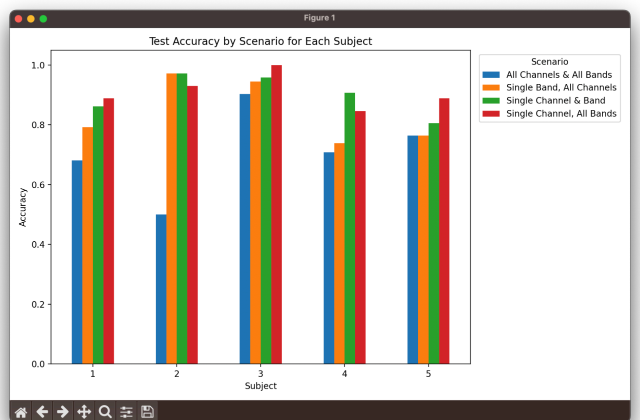
Subject	Scenario	Channel	Band	Best K	Accuracy
1	Single Band, All Channels	All	Delta	1	0.611111111111110
1	Single Band, All Channels	All	Theta	9	0.666666666666670
1	Single Band, All Channels	All	Alpha	7	0.791666666666670
1	Single Band, All Channels	All	Beta	5	0.736111111111110
1	Single Band, All Channels	All	Gamma	10	0.569444444444440
2	Single Band, All Channels	All	Delta	2	0.472222222222220
2	Single Band, All Channels	All	Theta	3	0.708333333333330
2	Single Band, All Channels	All	Alpha	7	0.972222222222220
2	Single Band, All Channels	All	Beta	1	0.833333333333330
2	Single Band, All Channels	All	Gamma	7	0.833333333333330
3	Single Band, All Channels	All	Delta	4	0.916666666666670
3	Single Band, All Channels	All	Theta	6	0.944444444444440
3	Single Band, All Channels	All	Alpha	3	0.861111111111110
3	Single Band, All Channels	All	Beta	8	0.527777777777780
3	Single Band, All Channels	All	Gamma	4	0.486111111111110
4	Single Band, All Channels	All	Delta	7	0.738461538461539
4	Single Band, All Channels	All	Theta	1	0.646153846153846
4	Single Band, All Channels	All	Alpha	3	0.615384615384615
4	Single Band, All Channels	All	Beta	2	0.538461538461538
4	Single Band, All Channels	All	Gamma	1	0.569230769230769
5	Single Band, All Channels	All	Delta	2	0.763888888888889
5	Single Band, All Channels	All	Theta	1	0.652777777777778
5	Single Band, All Channels	All	Alpha	5	0.555555555555556
5	Single Band, All Channels	All	Beta	2	0.652777777777778
5	Single Band, All Channels	All	Gamma	2	0.694444444444444

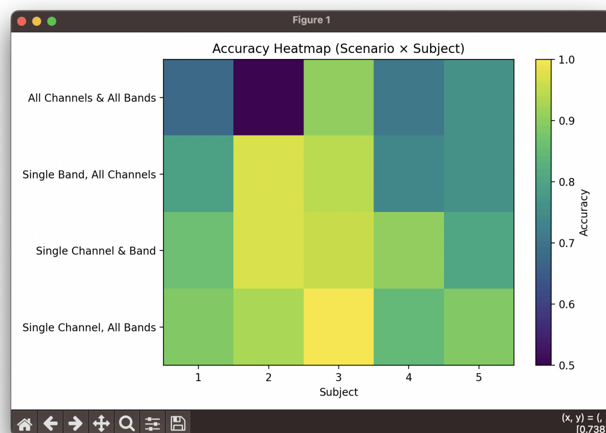
All channels, all bands

We combined all channels and all bads for a 7*5 matrix and we can see that it has a uniform performance drop for all subjects and that's probably because dimensionality causing overfitting in KNN.

Subject	Scenario	Channel	Band	Best K	Accuracy
1	All Channels & All Bands	All	All	7	0.680555555555560
2	All Channels & All Bands	All	All	1	0.5
3	All Channels & All Bands	All	All	7	0.902777777777780
4	All Channels & All Bands	All	All	7	0.707692307692308
5	All Channels & All Bands	All	All	2	0.763888888888889

We can also see this results from the graphs and diagrams plotted using the plotting program and we can see it follows the results and the comments correctly:





Comparison

Subjects 1 and 3 both peaked at C3 but subject 2 best channel was Pz, while subject 4 was F4 and subject 5 was C4. For the bands Alpha was the best accuray in subjects 1, 2, 4 suggesting that motor frontal alpha rhythms often carry the attention signal. While subject 3 was best classified by Theta and subject 5 by Delta perhaps reflecting individual differences in how vigilance vs drowsiness manifests spectrally. The K values ranged from just 2 neighbours for subject 3 up to 7 neighbours for subject 1 with subject 4 and 5 both at 5 and 2 at 3. And this shows the "smoothness" of the feature space differs across participants.

Conclusion

Our EEG-KNN pipeline demonstrated that optimal decoding parameters are highly subject-specific: the best electrode varied (C3 for Subjects 1 & 3, Pz for 2, F4 for 4, C4 for 5), the most informative frequency band ranged from Alpha in some to Theta or Delta in others, and the ideal neighborhood size K spanned 2 to 7. When we concatenated all five band-power features on a single channel, accuracy improved for some participants but degraded for others—showing that multi-band context can both enrich and obscure the signal. Aggregating one band across all channels generally reduced performance, suggesting spatial pooling often introduces noise rather than useful information. And flattening the full 7 × 5 feature matrix into one vector led to a uniform accuracy drop across subjects, a clear sign of the curse of dimensionality in KNN. Taken together, these results underscore the importance of tuning channel, band, and K on a per-subject basis or, if one must fuse features broadly, applying rigorous feature selection or dimensionality reduction to prevent overfitting.