```
% TRAINING CODE

clear all
close all

load('F.mat'); % Load the dataset F
load('N.mat'); % Load the dataset N
load('O.mat'); % Load the dataset O
load('S.mat'); % Load the dataset S
load('Z.mat'); % Load the dataset Z

% Training sets defined by the first 50 columns
F = F(:,1:50);
N = N(:,1:50);
O = O(:,1:50);
S = S(:,1:50);
Z = Z(:,1:50);

Fs = 173.61; % Define the sampling frequency
N_shift = length(F); % Set the length of the shifted dataset
% Define the frequency_shifted vector
frequencies_shifted = (linspace(-pi*Fs, Fs*(pi - (2*pi)/N_shift), N_shift) + (Fs*pi)/(N_shift)*mod(N_shift, 2))';

% Define the lower and higher indicies to filter
% These indicies will filter out signals lager than 120 Hz
lower_filter = 1600;
higher_filter = 2499;

% Find the shifted Fourier Transform
ffft = fft(F);
ffft = fftshift(ffft);
% Filter out frequencies higher than 120 Hz
ffft_filtered = zeros(size(ffft));
for k=1:size(ffft,2)
    ffft_filtered(:,k) = Rangefinder(ffft(:,k),lower_filter,higher_filter);
end
ffft = ffft_filtered;

% Find the shifted Fourier Transform
nfft = fft(N);
nfft = fftshift(nfft);
% Filter out frequencies higher than 120 Hz
nfft_filtered = zeros(size(nfft));
for k=1:size(nfft,2)
    nfft_filtered(:,k) = Rangefinder(nfft(:,k),lower_filter,higher_filter);
end
nfft = nfft_filtered;

% Find the shifted Fourier Transform
offt = fft(O);
offt = fftshift(offt);
% Filter out frequencies higher than 120 Hz
offt_filtered = zeros(size(offt));
for k=1:size(offt,2)
    offt_filtered(:,k) = Rangefinder(offt(:,k),lower_filter,higher_filter);
end
offt = offt_filtered;

% Find the shifted Fourier Transform
sfft = fft(S);
sfft = fftshift(sfft);
% Filter out frequencies higher than 120 Hz
```

```
sfft_filtered = zeros(size(sfft));
for k=1:size(sfft,2)
    sfft_filtered(:,k) = Rangefinder(sfft(:,k),lower_filter,higher_filter);
end
sfft = sfft_filtered;

% Find the shifted Fourier Transform
zfft = fft(Z);
zfft = fftshift(zfft);
% Filter out frequencies higher than 120 Hz
zfft_filtered = zeros(size(zfft));
for k=1:size(zfft,2)
    zfft_filtered(:,k) = Rangefinder(zfft(:,k),lower_filter,higher_filter);
end
zfft = zfft_filtered;

% Make a matrix with all the data
data = [sfft,nfft,offt,ffft,zfft];
% Find the principle component analysis for the matrix defined by all the
% data
[U,SS,VV] = svd(data,'econ');
% Find the linear combination of training data into a new basis set defined
% by the eigenvectors of the data
train_weights = U' * data;

% Averaging filter applied to the eigenvectors
for l=1:size(U,2)
    U(:,l) = movmean(U(:,l),7);
end
```

TEST CODE

```
load('F.mat'); % Load the dataset F
load('N.mat'); % Load the dataset N
load('O.mat'); % Load the dataset O
load('S.mat'); % Load the dataset S
load('Z.mat'); % Load the dataset Z

% Test sets defined by the last 50 columns
F2 = F(:,51:end);
N2 = N(:,51:end);
O2 = O(:,51:end);
S2 = S(:,51:end);
Z2 = Z(:,51:end);

% Find the shifted Fourier Transform
ffft = fft(F2);
ffft = fftshift(ffft);
% Filter out frequencies higher than 120 Hz
ffft_filtered = zeros(size(ffft));
for k=1:size(ffft,2)
    ffft_filtered(:,k) = Rangefinder(ffft(:,k),lower_filter,higher_filter);
end
ffft = ffft_filtered;

% Find the shifted Fourier Transform
nfft = fft(N2);
nfft = fftshift(nfft);
% Filter out frequencies higher than 120 Hz
nfft_filtered = zeros(size(nfft));
for k=1:size(nfft,2)
    nfft_filtered(:,k) = Rangefinder(nfft(:,k),lower_filter,higher_filter);
```

```
end
nfft = nfft_filtered;

% Find the shifted Fourier Transform
offt = fft(O2);
offt = fftshift(offt);
% Filter out frequencies higher than 120 Hz
offt_filtered = zeros(size(offt));
for k=1:size(offt,2)
    offt_filtered(:,k) = Rangefinder(offt(:,k),lower_filter,higher_filter);
end
offt = offt_filtered;

% Find the shifted Fourier Transform
sfft = fft(S2);
sfft = fftshift(sfft);
% Filter out frequencies higher than 120 Hz
sfft_filtered = zeros(size(sfft));
for k=1:size(sfft,2)
    sfft_filtered(:,k) = Rangefinder(sfft(:,k),lower_filter,higher_filter);
end
sfft = sfft_filtered;

% Find the shifted Fourier Transform
zfft = fft(Z2);
zfft = fftshift(zfft);
% Filter out frequencies higher than 120 Hz
zfft_filtered = zeros(size(zfft));
for k=1:size(zfft,2)
    zfft_filtered(:,k) = Rangefinder(zfft(:,k),lower_filter,higher_filter);
end
zfft = zfft_filtered;

% Define matrix for the test data
test = [sfft,ffft,nfft,offt,zfft];
% Project the test vectors into the eigenspace
test_weights = U' * test;

counter = 0;
for l=1:length(test_weights(1,:))
    % Find Euclidean distance to find nearest image
    [dist,index] = min(vecnorm(test_weights(:,l) - train_weights));
    % Check if the test vector is correctly identified as a seizure
    if l <= 50 && (1 <= index) && (index <= 50)
        counter = counter + 1;
    % Check if the test vector is correctly identified as a seizure
    elseif (51 <= l) && (51 <= index)
        counter = counter + 1;
    end
end
% Compute the accuracy of the algorithm
accuracy = counter / length(test_weights(:,1)) * 100
```

```
accuracy =

    80
```

Published with MATLAB® R2019a