```
% Step one: Convert CSV files to a single .mat
clear
clc
close all

error("Uncomment this line to read csv files again.")

DS_test_train_split = 0.9;

files_list = dir('../DS_csv/*.csv');

DS_split_threshold=floor(DS_test_train_split*size(files_list,1));


for i=1:size(files_list,1)

    temp_data=load(['./../DS_csv/' files_list(i).name]);
    % Normalize Stream data
    temp_markers=categorical(temp_data(:,5));
    %temp_data(:,5:6)=[];
    temp_data(temp_data>1000)=1000;
    temp_data(temp_data<-1000)=-1000;
    temp_data = temp_data/1000;

    if (i>DS_split_threshold)
        test_streams(i-DS_split_threshold).name = files_list(i).name;
        test_streams(i-DS_split_threshold).data = temp_data;
        test_streams(i-DS_split_threshold).markers = temp_markers;
    else
        train_streams(i).name = files_list(i).name;
        train_streams(i).data = temp_data;
        train_streams(i).markers = temp_markers;
    end
end

save('AllData.mat', 'train_streams',"test_streams");
```

```
% We only need to read from AllData.mat from now on
clear
clc
close all
load('AllData.mat', 'train_streams',"test_streams");

%F = 256; % Sampling frequency
%a=test_streams(1).data(:,1:4);
%audiowrite('test.wav',a,F);

Fs=256;
markers = categories(test_streams(1).markers);
markers (markers == categorical("0")) = [];

% Defining "Big Window"
```

```
pre_buff=200; % Number of samples taken as positive before marker
post_buff=500; % Number of samples taken as positive after marker

%Frame (is same as a Window if no stride used
frame_size = pre_buff+post_buff;
```

Extract known commands for Test

```
mkdir Dataset
mkdir Dataset/Test
DS_test_dir = './Dataset/Test/';
marked_range_test=zeros([size(test_streams,2),1000,2]);


for ss=1:size(test_streams,2)
    for mm=1:size(markers,1)
        % Find all occurances
        mrt_cnt=1;
        marker_locs = find(test_streams(ss).markers==markers(mm));
        %disp (size(marker_locs,1))

        % Loop over found markers
        for ii=1:size(marker_locs,1)

            marker_cell = markers(mm);
            marker_string = marker_cell{1};
            to_save_dir = [DS_test_dir marker_string '/'];

            if ~isfolder(to_save_dir)
                mkdir (to_save_dir)
            end

            to_save_name = [marker_string , '/' , num2str(ss) , ...
                                    '_' , num2str(ii) , '.wav'];
            to_save_file = [DS_test_dir to_save_name];

            ml = marker_locs(ii);
            if ml < pre_buff || ml > (size(test_streams(ss).data,1)-frame_size)
                continue
            end

            to_save_data = test_streams(ss).data(marker_locs(ii)-...
                                pre_buff:marker_locs(ii)+post_buff,:);


            marked_range_test(ss,mrt_cnt,:)=[marker_locs(ii)-...
                                pre_buff,marker_locs(ii)+post_buff];
            mrt_cnt=mrt_cnt+1;
            audiowrite(to_save_file, to_save_data,Fs,'BitsPerSample',32);
```

```
        end

    end
end
```

Extract known commands for Train

```
mkdir Dataset/Train
DS_train_dir = './Dataset/Train/';
marked_range_train=zeros([size(train_streams,2),1000,2]);


for ss=1:size(train_streams,2)
    for mm=1:size(markers,1)
        % Find all occurances

        mrT_cnt=1;

        marker_locs = find(train_streams(ss).markers==markers(mm));
        %disp (size(marker_locs,1))

        % Loop over found markers
        for ii=1:size(marker_locs,1)

            marker_cell = markers(mm);
            marker_string = marker_cell{1};

            to_save_dir = [DS_train_dir marker_string '/'];

            if ~isfolder(to_save_dir)
                mkdir (to_save_dir)
            end

            ml = marker_locs(ii);
            if ml < pre_buff || ml > (size(train_streams(ss).data,1)-frame_size)
                continue
            end

            to_save_name = [marker_string , '/' , num2str(ss) , ...
                                        '_' , num2str(ii) , '.wav'];
            to_save_file = [DS_train_dir to_save_name];

            to_save_data = train_streams(ss).data(marker_locs(ii)-...
                                pre_buff:marker_locs(ii)+post_buff,:);

            marked_range_train(ss,mrT_cnt,:)=[marker_locs(ii)-...
                                pre_buff,marker_locs(ii)+post_buff];
            mrT_cnt=mrT_cnt+1;

            audiowrite(to_save_file, to_save_data,Fs,'BitsPerSample',32);

        end
```

```
    end
end
```

Generate Background Noise

```
marked_range_train(:,mrT_cnt+5:end,:)=[];
marked_range_test (:,mrt_cnt+5:end,:)=[];

rng(100);

% Number of bg samples to take from each stream
bg_count_train=200;
bg_count_test=100;

% Background Noise for Test
for ss=1:size(test_streams,2)
    for bg=1:bg_count_test
        stream_length = size(test_streams(ss).data,1);

        % Search for a random and valid range to save as bg noise
        while 1

            rand_frame_start = randi([frame_size stream_length]);
            rand_frame_end   = rand_frame_start+frame_size;
            if rand_frame_end >= stream_length
                continue
            end
            is_range_valid = 1;

            for mr=1:size(marked_range_test,2)
                start_is_in_range = rand_frame_start>marked_range_test(ss,mr,1) && ...
                                    rand_frame_start<marked_range_test(ss,mr,1);

                end_is_in_range =   rand_frame_end>marked_range_test(ss,mr,2) && ...
                                    rand_frame_end<marked_range_test(ss,mr,2);

                if (start_is_in_range || end_is_in_range)
                    is_range_valid = 1;
                end
            end

            if (is_range_valid)
                break
            end
        end

        % Range found : [rand_frame_start, rand_frame_end]

        to_save_dir = [DS_test_dir 'BG/'];
```

```
        if ~isfolder(to_save_dir)
            mkdir (to_save_dir)
        end

        to_save_name = ['BG/' , num2str(ss) , ...
                                    '_' , num2str(bg) , '.wav'];
        to_save_file = [DS_test_dir to_save_name];

        to_save_data = test_streams(ss).data(rand_frame_start: rand_frame_end,:);
        audiowrite (to_save_file, to_save_data, Fs);
    end
end

% Background Noise for Train
for ss=1:size(train_streams,2)
    for bg=1:bg_count_train
        stream_length = size(train_streams(ss).data,1);

        % Search for a random and valid range to save as bg noise
        while 1

            rand_frame_start = randi([frame_size stream_length]);
            rand_frame_end   = rand_frame_start+frame_size;

            if rand_frame_end >= stream_length
                continue
            end

            is_range_valid = 1;

            for mr=1:size(marked_range_train,2)
                start_is_in_range = rand_frame_start>marked_range_train(ss,mr,1) && ...
                                    rand_frame_start<marked_range_train(ss,mr,1);

                end_is_in_range =   rand_frame_end>marked_range_train(ss,mr,2) && ...
                                    rand_frame_end<marked_range_train(ss,mr,2);

                if (start_is_in_range || end_is_in_range)
                    is_range_valid = 0;
                end
            end

            if (is_range_valid)
                break
            end
        end

        to_save_dir = [DS_train_dir 'BG/'];

        if ~isfolder(to_save_dir)
            mkdir (to_save_dir)
        end

        % Range found : [rand_frame_start, rand_frame_end]
```

```
        to_save_name = ['BG/' , num2str(ss) , ...
                                    '_' , num2str(bg) , '.wav'];
        to_save_file = [DS_train_dir to_save_name];

        to_save_data = train_streams(ss).data(rand_frame_start: rand_frame_end,:);
        audiowrite (to_save_file, to_save_data, Fs);
    end
end
```