Snore Detection using Deep Neural Networks

Team members

1) Sumitra devi Polina - 11555601

2) Sravanthi Tummala - 11533397

3) Devi Karri - 11543834

4) Naimisha Mummaneni - 11515092

Abstract:

We introduce a snore detection algorithm based on the combination of a convolutional neural network (CNN) and a recurrent neural network (RNN). Here we are obtaining audio recordings of subjects referred to a clinical center for a sleep study.
Snoring, a form of sleep-disordered breathing, interferes with sleep quality and quantity, both for the person who snores and often for the person who sleeps with the snorer. Poor sleep caused by snoring can create significant physical, mental, and economic problems.The goal of this study is to look into a way for analyzing snore sounds in order to detect the existence of various difficulties that a person may be experiencing. It could be harmless, but it could also be a sign of obstructive sleep apnea (OSA), a common sleep problem. Snoring detection that is accurate may aid in the screening and diagnosis of OSA.

Introduction:

Snoring is a pervasive peculiarity. It very well might be harmless, however, can likewise be a side effect of obstructive sleep apnea (OSA) , a common rest issue. The exact location of wheezing may assist with screening and determination of OSA. Techniques: We present a snore detection calculation in light of the blend of a convolutional neural network (CNN) and a recurrent neural network (RNN).

1. Problem specification:

Snore detection is difficult to find, so here we go to detect snores using python language and analyze our results with accurate accuracy.

1.1 Dataset

- To work on that, we used a sample snore dataset.
- When multiplied with a real spindle in the dataset, results in a high Amplitude signal wrt the Spindle.

1.2 Problem analysis

Sleep spindles are rhythmic brain activity patterns that can be seen in human electroencephalography (EEG) signals. The waveforms of these transitory patterns are often depicted as almost sinusoidal patterns in waves. In the community, sleep spindles have a wide range of properties (amplitude, duration, density), yet they are rather consistent for individuals.

2 Design and Milestones

The goal of this study is to look into a way for analyzing snore sounds in order to detect the existence of various difficulties that a person may be experiencing.

Data collection:

Data is collected from different sources. Currently we have used the data from the Kaggle, which are Ballistocardiography readings

2.1 Proposed Method :

We are using the Morlet's Wavelet for the convolving method.

Morlet's Wavelet for convolving:

The complex Morlet wavelet is convolved with the time-series signal, and the result of convolution is a complex-regarded signal from which flashing power and stage can be taken out at each time point.

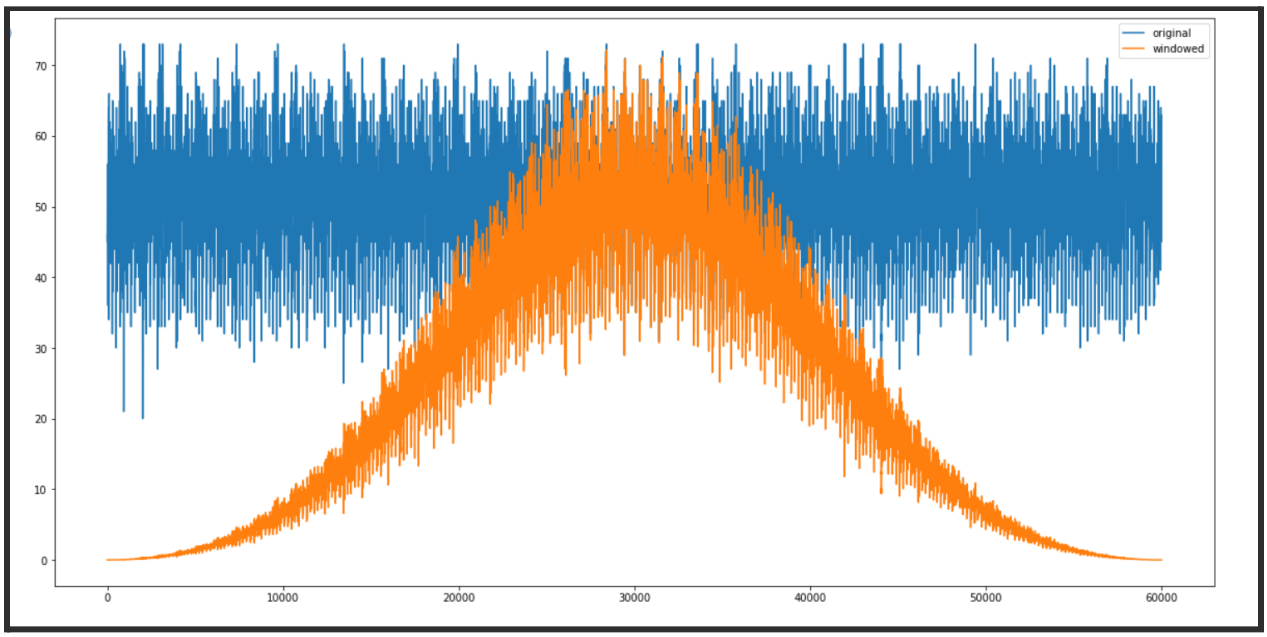
```
# Convolve the wavelet and extract magnitude and phase
def Wavelet_Formation(df, threshold, window_1, window_2):
    """
    Function to generate comparative graphs for spindle detection and analysis.

    Parameters:
    -----------
    df: pd.Dataframe, Contains Ballistography Data.
    threshold:  float, Act as cut-off factor on Normalized Wavelet.
    window_1: int, starting Time stamp in the Graph
    window_1: int, Ending Time stamp in the Graph

    Returns:
    --------

    Graph-1: 1). Amplitude V/S Time-Stamps 2). Differenced Series Amplitude V/S Time-Stamps.
    Graph-2: Normalized Wavelet Power(cutoff set by threshold) V/S Time-Stamps.
    """
    df['FirstOrderDiff'] = df['Amplitude'].diff().bfill()
    analytic = np.convolve(df.FirstOrderDiff, wlt, mode='same')
    magnitude = np.abs(analytic)
    phase = np.angle(analytic)
    # Square and normalize the magnitude from 0 to 1 (using the min and max)
        # Square and normalize the magnitude from 0 to 1 (using the min and max)
    power = np.square(magnitude)
    norm_power = (power - power.min()) / (power.max() - power.min())
        # Define the threshold
    thresh = threshold
        # Find supra-threshold values
    supra_thresh = np.where(norm_power >= thresh)[0]
```

2.2 Data processing:

```
test1 = pd.read_csv('sample1.csv', header = None, names = ['Amplitude'])
test2 = pd.read_csv('sample2.csv', header = None, names = ['Amplitude'])
```

Windowing :

The windowing procedure incorporates expanding the best inspiration response with a window ability to make a related channel, which fixes the best drive response. Like the repeat looking at the method, the windowing methodology makes a channel whose repeat response approximates an optimal repeat response.



Tuning:

- Tuning is the most common way of amplifying a model's exhibition without overfitting or making excessively high fluctuations.
- Spindle Frequency is reliant upon Age, so tuning is generally required in light of Patient/Sample.

2.3 Experimental Settings :

- Feature engineering is the most widely recognized approach to picking and changing elements while making a farsighted model using AI. It's a fair technique for working on farsighted models as it incorporates detaching key information, including models, and getting someone with region dominance.

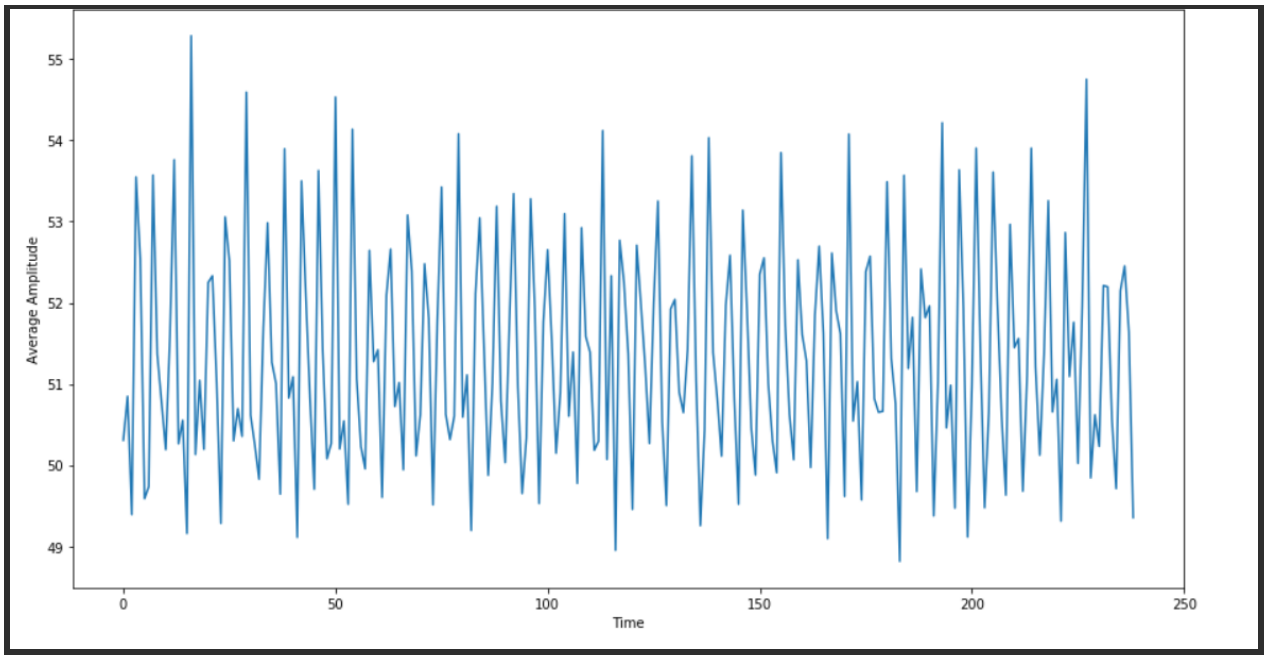
```python
def create_features(df):
    """Function to create the following features:
    1. Avg_Amplitude :  Returns Average Amplitude per 250 Time Stamps.
    2. Min_Amplitude : Returns Minimum Amplitude per 250 Time Stamps.
    3. Max_Amplitude : Returns Maximum Amplitude per 250 Time Stamps.
    4. StDev_Amplitude: Returns Standard Deviation of Amplitude per 250 Time Stamps.
    5. Seconds: Returns value in Seconds ie., between 1-60.
    6. Minute: Returns the n-th minute for a particular record.
    Parameters:
    ----------
    df: pd.DataFrame, Dataset containing Body Vibrations.

    Returns:
    --------
    df2: pd.DataFrame, Dataset containing Added Features.

    Assumptions:
    ------------
    # Assumption-1 : 250 Time-stamps == 1 Second
    # Assumption-2 : Considering only 3 minutes and 59 seconds. Therefore, 59750 Time-stamps.
    """

    df2 = pd.DataFrame(data=None)
    array = df.Amplitude[0:59750].values.reshape(-1,250)
    df2['Avg_Amplitude'] = [np.mean(i) for i in array]
    df2['Min_Amplitude'] = [np.min(i) for i in array]
    df2['Max_Amplitude'] = [np.max(i) for i in array]
    df2['StDev_Amplitude'] = [np.std(i) for i in array]
    df2['Seconds'] = np.tile([int(i) for i in range(1,61)], array.shape[0])[0:array.shape[0]]
    df2['Minute'] = np.repeat([minute for minute in ['1st', '2nd', '3rd','4th']], 60)[0:array.shape[0]]
    return df2
```

2.4 Validation methods :

	Amplitude	FirstOrderDiff
count	59993.000000	59992.000000
mean	51.344357	0.000150
std	7.377195	1.895782
min	20.000000	-12.000000
25%	47.000000	-1.000000
50%	52.000000	0.000000
75%	56.000000	1.000000
max	73.000000	12.000000

Plotting Average Amplitude

The graph is plotted between time and average amplitude.



Limitations :

The major limitation is that doing snore detection, other voices cause problems like fan sound etc.

Future Work :

- We are going to work on spectral analysis and the frequency domain filter.
- In future, we can further work on it and also can introduce this in the market for many useful purposes.

References :

https://www.sciencedirect.com/science/article/pii/S0169260720317508

Github link:

Sravanthi2706/Snore-Detection (github.com)