Aufgabe 1: Merkmalsextraktion I (Fensterung)

In dieser Übung soll der erste Teil der Merkmalsextraktion für den Spracherkenner implementiert werden. Entpacken Sie den Inhalt des Archivs (verfügbar in ISIS) für die aktuelle Übung in die lokale Kopie Ihres Repositories. Dieses sollte nun folgende Struktur aufweisen:

```
./my-repository/recognizer/__init__.py
./my-repository/recognizer/feature_extraction.py
./my-repository/recognizer/tools.py
./my-repository/data/TEST-MAN-AH-3O33951A.wav
./my-repository/uebung1.py
```

Öffnen Sie anschließend den Pfad zu Ihrem lokalen Repository in der Konsole und aktualisieren Sie das Remote-Repository mit den entsprechenden Befehlen:

```
~/path-to-my-repository$ git pull
~/path-to-my-repository$ git add --all
~/path-to-my-repository$ git commit -m "Uebung 1 hinzugefuegt."
~/path-to-my-repository$ git push
```

Implementierung der benötigten Hilfsfunktionen

Für die Merkmalsextraktion werden einige Hilfsfunktionen benötigt, die in diesem Teil der Übung implementiert werden sollen. Für diese und alle weiteren Übungen gilt, dass sämtliche Hilfsfunktionen in der Datei `tools.py` der `recognizer` Bibliothek implementiert werden müssen, da diese teilweise übungsübergreifend verwendet werden.

Hinweis: Bitte achten Sie grundsätzlich auf die Ein- und Ausgabeparameter. Falsche Funktionsköpfe sowie eine falsche Reihenfolge der Dimensionen führen automatisch zu null Punkten.

1.1 Implementieren Sie eine Funktion

```
def sec_to_samples(x, sampling_rate):
    ...
```

in der Datei `tools.py`, die einen skalaren Zeitwert `x` in Sekunden für eine gegebene Abtastfrequenz `sampling_rate` in Hz in den entsprechenden diskreten Abtastzeitpunkt umwandelt und diesen Wert im Datentyp `int` zurück gibt.

1.2 Implementieren Sie eine Funktion

```
def next_pow2(x):
    ...
```

in der Datei `tools.py`, die für einen Zahlenwert `x` die nächstgrößere Zweierpotenz im Datentyp `int` zurück gibt. Die Funktion soll z. B. bei `x = 300` eine `9` zurück geben.

Schauen Sie sich hierzu bei Bedarf auch die entsprechende Funktion in Matlab an (`https://de.mathworks.com/help/matlab/ref/nextpow2.html`).

1.3 Implementieren Sie eine Funktion

```
def dft_window_size(x, sampling_rate):
    ...
```

in der Datei `tools.py`, die für eine Länge `x` in Sekunden und die Abtastfrequenz `sampling_rate` in Hz die Länge als Anzahl von Abtastwerten im Datentyp `int` zurück gibt und dazu die nächstgrößere Zweierpotenz wählt. Nutzen Sie hierfür die vorher implementierten Funtionen `sec_to_samples()` und `next_pow2()`.

1.4 Implementieren Sie eine Funktion

```
def get_num_frames(signal_length_samples, window_size_samples,
                   hop_size_samples):
    ...
```

in der Datei `tools.py`, die für eine diskrete Signallänge `signal_length_samples` (entspricht der Gesamtanzahl von Abtastzeitpunkten in einem Signal), eine Fensterlänge `window_size_samples` und einen Rahmenvorschub `hop_size_samples` (beides ebenfalls angegeben als Anzahl von diskreten Abtastzeitpunkten) die Anzahl der benötigten Rahmen im Datentyp `int` zurück gibt. Verwenden Sie zur Berechnung den Zusammenhang

$$K = \left\lceil \frac{Q - O}{R} \right\rceil,$$

wobei K die gesuchte Anzahl der Rahmen bezeichnet, Q die Anzahl der Abtastzeitpunkte im Signal ist, N die Rahmenlänge ist, R der Rahmenverschub ist und O die Anzahl der überlappenden Abstatstzeitpunkte zwischen zwei aufeinanderfolgenden Rahmen beschreibt, mit $O = N$ - R. Das Ergebnis muss entsprechend auf die nächsthöhere ganze Zahl aufgerundet werden.

Implementierung der Fensterung

Im ersten Teil der Merkmalsextraktion soll eine Fensterung durchgeführt werden. Hierzu wird in dieser Übung zunächst die Funktion für die Fensterung in der Datei `feature_extraction.py` der `recognizer` Bibliothek implementiert. Verwenden Sie hierbei auch die zuvor implementierten Hilfsfunktionen.

1.5 Implementieren Sie eine Funktion

```
def make_frames(audio_data, sampling_rate, window_size, hop_size):
    ...
```

in der Datei `feature_extraction.py`, die ein einkanaliges Audiosignal in Form eines Arrays `audio_data` mit der Abtastfrequenz `sampling_rate` in Hz für eine gegebene Fensterlänge `window_size` und Rahmenvorschub `hop_size` (beide angegeben in Sekunden) in überlappende Rahmen gleicher Länge einteilt. Die Fensterlänge soll hierbei zunächst so modifiziert werden, dass sie im zeitdiskreten Bereich der nächsthöheren

Zweierpotenz entspricht. Jeder Rahmen soll anschließend zusätzlich mit einem dieser Fensterlänge entsprechenden Hamming-Fenster multipliziert werden. Die Funktion soll das Ergebnis als zweidimensionales Array im Datentyp `float` zurückgeben, wobei die erste Arraydimension der Anzahl der Rahmen und die zweite Arraydimension der modifizierten Fensterlänge entsprechen muss. Darüber hinaus muss im letzten Rahmen ein Zero-Padding durchgeführt werden, falls die Länge des Audiosignals nicht ausreichend ist.

Hilfreiche Funktionen: `numpy.hamming()`, `numpy.pad()`

Einlesen einer Audiodatei

1.6 Öffnen Sie die Datei `uebung1.py` und schreiben Sie hier ein Skript, welches die Audiodatei `TEST-MAN-AH-3O33951A.wav` im Unterordner `data` einliest und die zuvor implementierte Funktion `make_frames()` aufruft. Verwenden Sie hierbei eine Fensterlänge von 25 ms und einen Rahmenvorschub von 10 ms.

Hilfreiche Funktionen: `scipy.io.wavfile.read()`

1.7 Stellen Sie die ersten vier Frames in Form von Subplots graphisch dar. Die x-Achse sollte hierbei in Sekunden angegeben werden. Fügen Sie auch eine entsprechende Achsenbeschriftung hinzu. Das Ergebnis sollte in etwa so wie in Abbildung 2 aussehen. Verwenden Sie hierbei die Parameter `window_size`= 0.4 und `hop_size`= 0.25. Die Abbildung soll Ihnen den gewählten Rahmenvorschub und die Fensterung veranschaulichen. Nutzen Sie `matplotlib.pyplot.show()`, damit der Plot angezeigt wird. **Hinweis:** Eventuell ist es notwendig, PyQt5 via `pip install PyQt5` zu installieren.

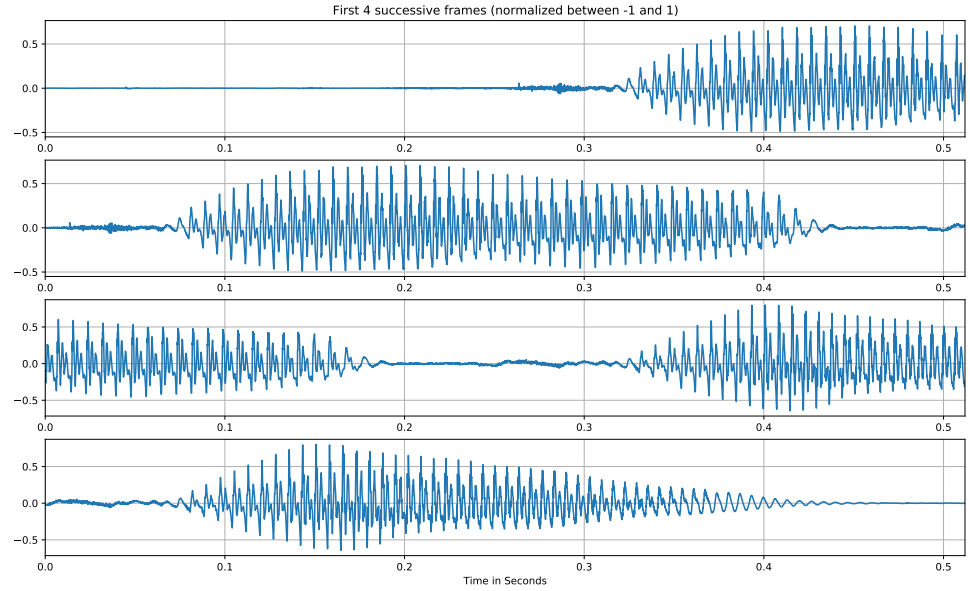

Figure 1: Dastellung der ersten vier Frames (ohne Multiplikation mit einem Hamming-Fenster) für `TEST-MAN-AH-3033951A.wav`

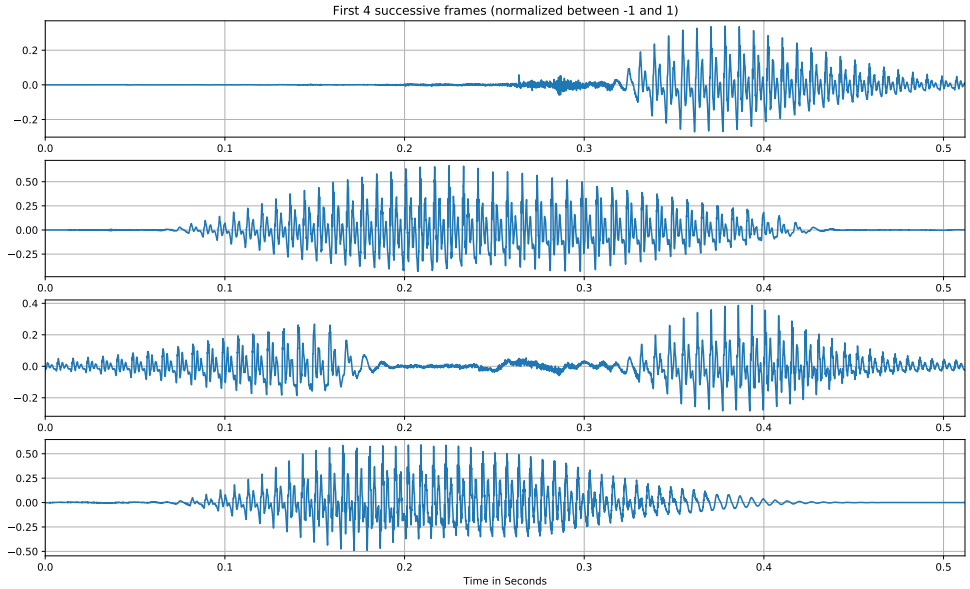

Figure 2: Dastellung der ersten vier Frames (mit Multiplikation mit einem Hamming-Fenster) für `TEST-MAN-AH-3033951A.wav`