CS574 Assignment 4 Improvements

Speech separation using visual and speech cues

Group 15:

Shubham Goel	160101083
Shaurya Gomber	160101086
Archit Jugran	160101087
Rishabh Jain	160101088
Shashwat Jolly	160123036

Introduction

1. Base implementation : “Looking to Listen at the Cocktail Party: A Speaker-Independent Audio-Visual Model for Speech Separation” by Ephrat et. Al. 2018

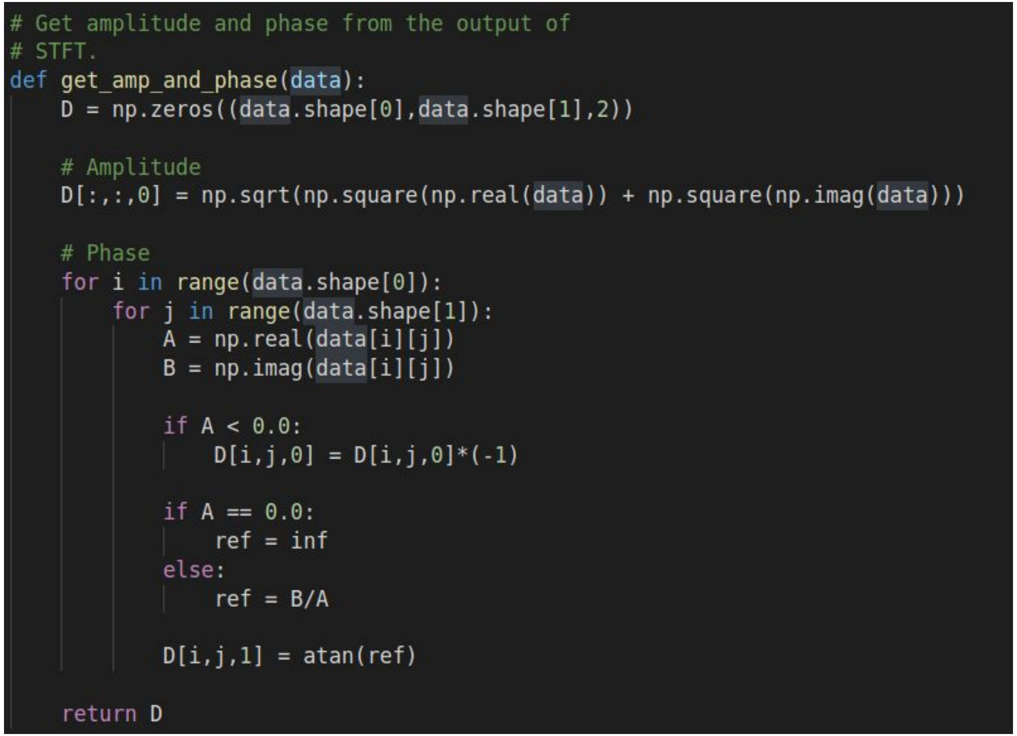
2. All the different models (after different modifications) were trained on a training set with around 100 examples for 40 epochs.

3. We use Short Term Objective Intelligibility (stoi) measures to measure the correlation (similarity) between the target audio and the enhanced mixed audio.

4. The base implementation was trained as per the conditions mentioned above and a **stoi similarity of 68.74%** was observed on the testing set.

Modifications Implemented

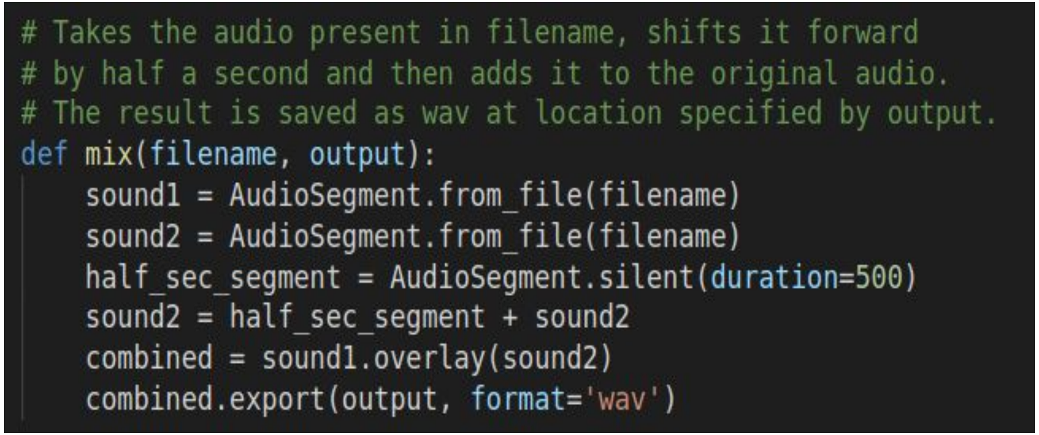
Predicting True Phase



```python
# Get amplitude and phase from the output of
# STFT.
def get_amp_and_phase(data):
    D = np.zeros((data.shape[0],data.shape[1],2))

    # Amplitude
    D[:,:,0] = np.sqrt(np.square(np.real(data)) + np.square(np.imag(data)))

    # Phase
    for i in range(data.shape[0]):
        for j in range(data.shape[1]):
            A = np.real(data[i][j])
            B = np.imag(data[i][j])

            if A < 0.0:
                D[i,j,0] = D[i,j,0]*(-1)

            if A == 0.0:
                ref = inf
            else:
                ref = B/A

            D[i,j,1] = atan(ref)

    return D
```

- Parameter data is a 298 **x** 257 matrix.
- Each $(i,j)^{th}$ value corresponds to the i^{th} window (STFT) and the j^{th} frequency bin in that window.
- Each $(i,j)^{th}$ value is a complex number (A+iB).
- Amplitude is calculated as $\sqrt{A^2 + B^2}$
- Phase is calculated as $\tan^{-1}(B/A)$.
- Stoi Similarity - **45.83%**

Noise Invariant Training



```python
# Takes the audio present in filename, shifts it forward
# by half a second and then adds it to the original audio.
# The result is saved as wav at location specified by output.
def mix(filename, output):
    sound1 = AudioSegment.from_file(filename)
    sound2 = AudioSegment.from_file(filename)
    half_sec_segment = AudioSegment.silent(duration=500)
    sound2 = half_sec_segment + sound2
    combined = sound1.overlay(sound2)
    combined.export(output, format='wav')
```

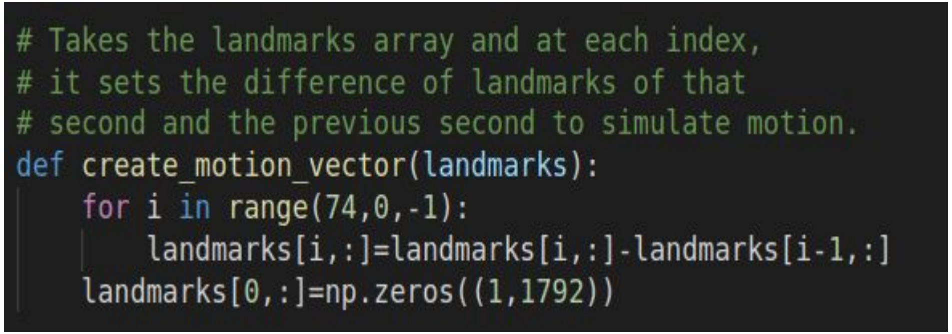
- Created training data where voice of the target speaker is added to the audio as noise.
- Shifted the audio forward by 0.5 seconds.
- Added this shifted audio to the original audio.
- Stoi Similarity - **71.95%**

Motion Vector

Landmarks is an array of size 75 **x** 1 **x** 1792

75 : number of frames

1792 : landmarks corresponding to face in the frame



```
# Takes the landmarks array and at each index,
# it sets the difference of landmarks of that
# second and the previous second to simulate motion.
def create_motion_vector(landmarks):
    for i in range(74,0,-1):
        landmarks[i,:]=landmarks[i,:]-landmarks[i-1,:]
    landmarks[0,:]=np.zeros((1,1792))
```

- Motion vector for the 1st second is set to all 0s.
- For each frame, the landmarks of the previous frame are subtracted from the landmarks of that frame.
- Stoi Similarity - **65.49%**

Noise Invariant Training with Motion Vector

- Motion vector is used for face embedding.
- Noise-invariant training samples are also added.
- Stoi Similarity - **69.23%**

Results

Modifications Implemented	Stoi Similarity Observed
None	68.74%
Predicting True Phase	45.83%
Noise-Invariant Training	**71.95%**
Motion Vector of Landmarks	65.49%
Noise-Invariant + Motion Vector	69.23%

Conclusions

1. Phase implementation didn't give good results. Reasons might be:
    - Magnitude and Phase are on different scales. So, different models should be tried to predict both of them separately.
    - Approximations and loss of precision when using sqrt, atan etc.
2. Adding Noise-Invariant training makes the model perform better than the base implementation.
3. Adding motion vector of landmarks gives good results but not better than base implementation. So does, adding noise-invariant and motion vector together.
4. All different modifications were trained on 100 videos for 40 epochs and the results mentioned here are for this setting only. The results and comparisons might change for different settings.