**comp\_snr**

**function [snr\_mean, segsnr\_mean] = comp\_snr(cleanFile, enhdFile);**

**Purpose**: This function calculates the overall and segmental Signal-to-Noise Ratio (SNR) between a clean speech file and an enhanced speech file.

**Input**: Two arguments: cleanFile (the original clean audio file) and enhdFile (the enhanced audio file).

**Output**: Two outputs: snr\_mean (overall SNR) and segsnr\_mean (mean segmental SNR).

**if nargin ~= 2**

**fprintf('USAGE: [snr\_mean, segsnr\_mean]= comp\_SNR(cleanFile, enhdFile) \n');**

**return;**

**end**

**Check Input Arguments**: The function verifies that exactly two input arguments are provided. If not, it prints usage instructions and exits.

**[data1, Srate1] = audioread(cleanFile);**

**[data2, Srate2] = audioread(enhdFile);**

**Read Audio Files**: It reads the clean and enhanced audio files, extracting both the audio data (data1, data2) and their sample rates (Srate1, Srate2).

**if (Srate1 ~= Srate2)**

**error('The two files do not match!\n');**

**end**

**Sample Rate Check**: Ensures both files have the same sample rate, which is crucial for accurate comparison. If they don't match, an error is raised.

**len = min(length(data1), length(data2));**

**data1 = data1(1:len);**

**data2 = data2(1:len);**

**Length Adjustment**: Determines the shorter length of the two audio signals and trims both to that length, ensuring they are of equal length for further analysis.

**[snr\_dist, segsnr\_dist] = snr(data1, data2, Srate1);**

**SNR Calculation**: Calls a nested function snr to compute both overall and segmental SNR between the two signals.

**snr\_mean = snr\_dist;**

**segsnr\_mean = mean(segsnr\_dist);**

**Mean Calculation**: Assigns the overall SNR to snr\_mean and calculates the mean of the segmental SNR values to get segsnr\_mean.

**function [overall\_snr, segmental\_snr] = snr(clean\_speech, processed\_speech, sample\_rate)**

**Purpose**: This nested function computes both overall and segmental SNR values for the provided clean and processed speech signals.

**clean\_length = length(clean\_speech);**

**processed\_length = length(processed\_speech);**

**Length Check**: Retrieves the lengths of both clean and processed speech signals for verification.

**if (clean\_length ~= processed\_length)**

**disp('Error: Both Speech Files must be same length.');**

**return;**

**end**

**Length Verification**: Checks if both signals are of the same length. If not, it displays an error message and exits.

**overall\_snr = 10 \* log10(sum(clean\_speech.^2) / sum((clean\_speech - processed\_speech).^2));**

**Overall SNR Calculation**:

* **Signal Energy**: sum(clean\_speech.^2) calculates the total energy of the clean signal. The energy of a signal is the sum of the squares of its samples.
* **Noise Energy**: sum((clean\_speech - processed\_speech).^2) computes the energy of the noise (the difference between the clean and processed signals).
* **SNR Formula**: The SNR is given by SNR=10log10(Signal Energy / Noise Energy). This formula expresses the ratio of signal power to noise power in decibels (dB).

**winlength = round(30 \* sample\_rate / 1000); % 240**

**skiprate = floor(winlength / 4); % 60**

**Windowing Parameters**:

* winlength: Defines the length of the analysis window (30 ms). It is converted into samples based on the sample rate.
* skiprate: Determines how many samples to skip for the next frame, creating a 25% overlap between adjacent windows.

**MIN\_SNR = -10; % minimum SNR in dB**

**MAX\_SNR = 35; % maximum SNR in dB**

**Clamping Values**: Defines the minimum and maximum values for segmental SNR to prevent unrealistic values.

**num\_frames = clean\_length / skiprate - (winlength / skiprate);**

**Frame Calculation**: Computes the number of frames that will be processed based on the signal length and windowing parameters.

**start = 1; % starting sample**

**window = 0.5 \* (1 - cos(2 \* pi \* (1:winlength)' / (winlength + 1)));**

**Window Function**: Initializes a Hanning window to taper the edges of each frame, reducing spectral leakage during analysis.

**for frame\_count = 1:num\_frames**

**Frame Loop**: Iterates through each frame of the audio signals.

**clean\_frame = clean\_speech(start:start + winlength - 1);**

**processed\_frame = processed\_speech(start:start + winlength - 1);**

**clean\_frame = clean\_frame .\* window;**

**processed\_frame = processed\_frame .\* window;**

**Frame Extraction**: Extracts segments (frames) of the clean and processed signals and applies the Hanning window to them.

**signal\_energy = sum(clean\_frame.^2);**

**noise\_energy = sum((clean\_frame - processed\_frame).^2);**

**Frame Energy Calculation**:

* **Signal Energy**: Similar to overall SNR, calculates the energy for the current clean frame.
* **Noise Energy**: Calculates the energy for the difference between the clean and processed frames.

**segmental\_snr(frame\_count) = 10 \* log10(signal\_energy / (noise\_energy + eps) + eps);**

**Segmental SNR Calculation**:

* The same SNR formula is applied here for each frame.
* eps is a small constant added to avoid division by zero errors.

**segmental\_snr(frame\_count) = max(segmental\_snr(frame\_count), MIN\_SNR);**

**segmental\_snr(frame\_count) = min(segmental\_snr(frame\_count), MAX\_SNR);**

**Clamping Segmental SNR**: Ensures that the calculated segmental SNR is within the predefined limits.

**start = start + skiprate;**

**Frame Update**: Moves the starting sample for the next frame by the skip rate to ensure the next segment is analyzed.

**[SNRovl, SNRseg] = comp\_snr('/MATLAB Drive/Audio/voice\_long.wav', '/MATLAB Drive/Noise\_Audio/noisy\_voice\_long.wav');**

The output values you received, **SNRovl = 7.64 dB** and **SNRseg = 7.41 dB**, provide insights into the quality of the enhanced audio compared to the clean audio.

**Overall SNR (SNRovl)**

* **Value**: 7.64 dB
* **Interpretation**: This value represents the overall signal-to-noise ratio of the entire audio file. A value of 7.64 dB indicates that the clean signal (the original audio) is about 7.64 dB higher in power than the noise introduced in the enhanced signal.
  + **General Context**:
    - SNR values typically range from negative values (indicating more noise than signal) to positive values (indicating more signal than noise).
    - An overall SNR above 0 dB suggests that the enhanced signal retains some quality relative to the clean signal, but at 7.64 dB, the enhancement has only moderately improved the signal quality. Values between 5 dB and 10 dB are often considered to indicate acceptable quality but not ideal.

**Segmental SNR (SNRseg)**

* **Value**: 7.41 dB
* **Interpretation**: This value represents the mean segmental signal-to-noise ratio calculated across multiple frames of the audio. Like the overall SNR, it indicates that on average, the clean signal is about 7.41 dB higher in power than the noise for each segment analyzed.
  + **General Context**:
    - Segmental SNR provides a more detailed view of how the signal quality varies across different parts of the audio.
    - A segmental SNR around 7.41 dB indicates that the enhancement is consistent but not dramatically effective across different segments.

**[SNRovl, SNRseg] = comp\_snr('/MATLAB Drive/Audio/voice\_long.wav', '/MATLAB Drive/Pred\_Audio/knm\_00180\_00011382081\_8000Hz\_5db\_denoise1.wav');**

The SNR results you obtained after using a U-Net model to denoise audio show a significant drop in quality:

### Interpretation of Results

1. **Overall SNR (SNRovl = -5.16 dB)**:
   * A negative overall SNR indicates that the noise power is greater than the signal power. This means that, overall, the denoised output still has more noise than the original clean signal, which suggests that the U-Net model did not effectively reduce the noise in the audio.
2. **Segmental SNR (SNRseg = -4.37 dB)**:
   * Similarly, the negative segmental SNR means that, on average, the clean voice is still overshadowed by the noise across different segments of the audio. This implies that the noise is significantly affecting the intelligibility and quality of the audio.

### Contextual Understanding

* **Denoising Effectiveness**: The expectation from a denoising model is to improve the SNR values, ideally bringing them above 0 dB, indicating that the clean signal is more prominent than the noise.
* **Model Performance**: These negative values suggest that the model may not have learned effectively from the training data, or that the noise levels in the input files were too high for the model to handle successfully.