Final Project - 2019

Audio-Visual Speech Enhancement

Group 322

Group Members:

Matan Aharonov
Matan.Aharonov@mail.huji.ac.il

Yakov Monfred
Yakov.Monfred@mail.huji.ac.il

Advisor:

Mr. Aviv Gabbay
Aviv.Gabbay@mail.huji.ac.il
Computer Vision Lab, The Hebrew University of Jerusalem

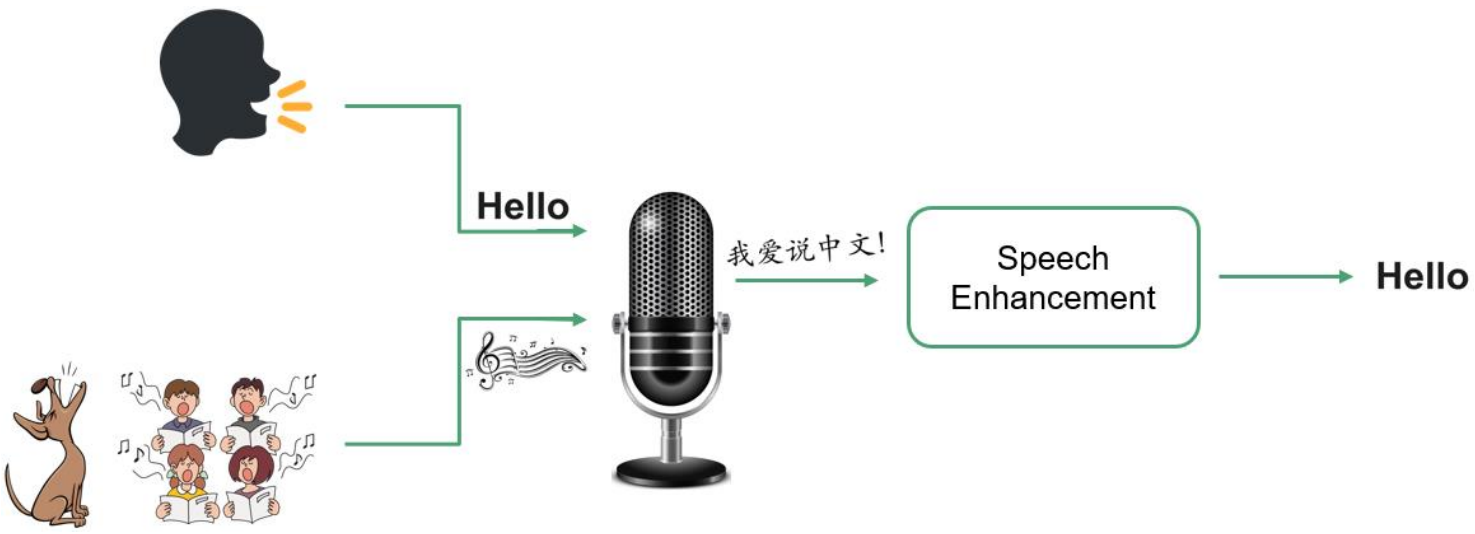
Table of Contents

Abstract

Speech enhancement aims to improve speech quality by using various algorithms.
When video is shot in noisy environment, the voice of a speaker seen in the video can be enhanced using the visible mouth movements, reducing background noise.

Isolating the voice of a specific person while filtering out other voices or background noises has been shown to be an extremely challenging task. Existing methods mostly rely on processing the auditory signals recorded by a single microphone alone. Other approaches require configuration of microphone arrays and are not feasible in most of the cases. Our project leverages the lip movements captured by a video camera to enhance the voice of the speaker of interest.

How convenient will it be if we mute all the noise around but the voice of the speaker? Given a video of a speaking person, along with a noisy soundtrack, we implemented an assistant enhancing the voice of the speaker while eliminating the background noise in real-time by a neural network-based model.



Introduction

Speech enhancement is designed to improve speech quality and intelligibility when audio is recorded in noisy environments. Applications include telephone conversations, video conferences, TV reporting and more. In hearing aids, speech enhancement can reduce discomfort and increase intelligibility. Speech enhancement can also be used as a preliminary step in speech recognition and speaker identification systems.

Speech enhancement has been the subject of extensive research, and has recently benefited from advancements in machine lip reading and speech reading.
While most existing methods use audio-only inputs, improved performance is obtained with our audio-visual end-to-end neural network model for separating the voice of a visible speaker from background noise.

In our solution, we model the correlation between facial mouth movements and auditory speech signals, and provide a system that processes both the video and audio signals captured by a camera and a microphone, and outputs an enhanced speech signal.
More specifically, the audio signal is decomposed into time-frequency components that are filtered accordingly by a neural network-based model.

Once the model is trained on a specific speaker, it can be used to enhance the voice of this speaker. We assume a video showing the face of the target speaker is available along with the noisy soundtrack, and use the visible mouth movements to isolate the desired voice from the background noise

Since our project is a real-time system, we needed to analyze a live video.
This analysis often resulted in a lengthy processing time which led to downgrading the user experience. To overcome this difficulty, we used parallel computation to speed the process time.

We evaluate the performance of our model in different speech enhancement experiments (English, Hebrew and ambient noises). We also show that this method can be integrated into real-time systems.

Related Work

Previous work has been done on speech enhancement. Previous methods include lip reading, and methods that mostly rely on processing the auditory signals recorded by a single microphone alone.

Lip Reading

Speech enhancement has recently benefited from advancements in machine lip reading [1]. Lip reading is the task of decoding text from the movement of a speaker's mouth. Traditional approaches separated the problem into two stages: designing or learning visual features, and prediction. More recent deep lipreading approaches are end-to-end trainable.

Audio-only deep learning-based speech enhancement

Previous methods for single-channel speech enhancement mostly use audio only input.
Lu et al. [2] train a deep auto-encoder for denoising the speech signal. Their model predicts a mel-scale spectrogram representing the clean speech. Pascual et al. [3] use generative adversarial networks and operate at the waveform level. Despite their decent overall performance, audio-only approaches achieve lower performance when separating similar human voices, as commonly observed in same-gender mixtures [4].

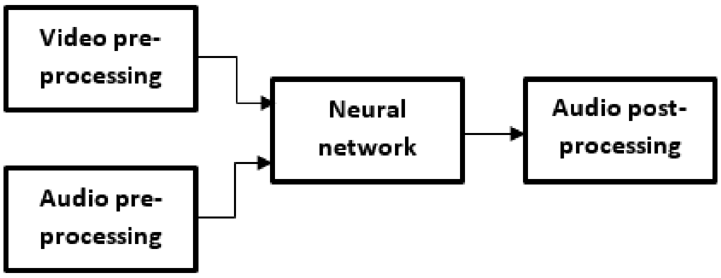
Audio-visual speech enhancement

Work has also been done on audio-visual speech enhancement and separation.
Kahn and Milner [5] use hand-crafted visual features to derive binary and soft masks for speaker separation. Hou et al. [6] propose convolutional neural network model to enhance noisy speech. Their network gets a sequence of frames cropped to the speaker's lips region and a spectrogram representing the noisy speech, and outputs a spectrogram representing the enhanced speech. Gabbay et al. [7] feed the video frames into a trained speech generation network [8], and use the spectrogram of the predicted speech to construct masks for separating the clean voice from the noisy input.

Method & Materials

Solution pipeline

We propose a speech enhancement by processing the two inputs (lips video, noisy-speech audio) in a single step manner (End-to-End speech enhancement).
The solution consists of the following parts: video and audio pre-processing, training and prediction using the neural network algorithm, and audio post-processing.



- **Video pre-processing**
  The video is divided to non-overlapping segments of 6 frames (200 ms) each.
  From every frame we crop a mouth-centered window of size 128×128 pixels, using the 20 mouth landmarks from the 68 facial landmarks file. The video segment used as input to the neural network is therefore of size 128×128×6. We normalize the video inputs over the entire training data by subtracting the mean video frame and dividing by the standard deviation.
- **Audio pre-processing**
  The corresponding audio signal is resampled to 16 kHz. Short-Time Fourier-Transform (STFT) is applied to the waveform signal. The spectrogram (STFT magnitude) is used as input to the neural network, and the phase is kept aside for reconstruction of the enhanced signal. We set the STFT window size to 640 samples, which equals to 40 milliseconds and corresponds to the length of a single video frame. We shift the window by hop length of 160 samples at a time, creating an overlap of 75%. Log mel-scale spectrogram is computed by multiplying the spectrogram by a mel-spaced filter bank. The log mel-scale spectrogram comprises 80 mel frequencies from 0 to 8000 Hz. We slice the spectrogram to pieces of length 200 milliseconds corresponding to the length of 6 video frames, resulting in spectrograms of size 80×20: 20 temporal samples, each having 80 frequency bins.

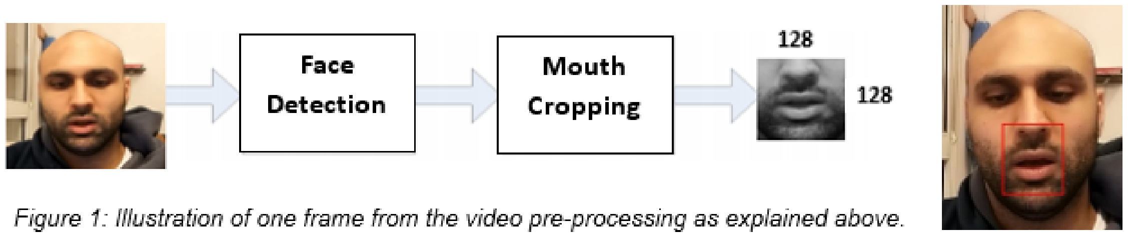


Figure 1: Illustration of one frame from the video pre-processing as explained above.

Algorithm details

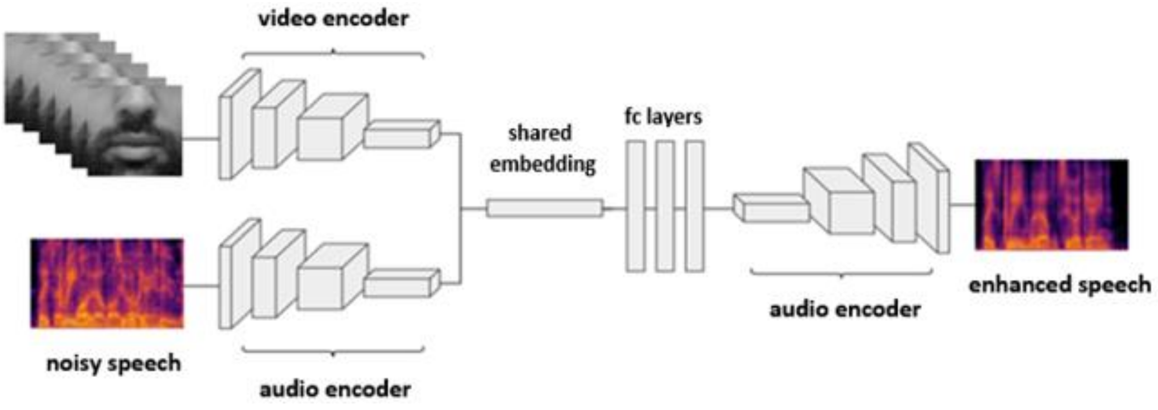


Figure 2: Illustration of our encoder-decoder model architecture [9]. A sequence of 6 video frames centered on the mouth region is fed into a convolutional neural network creating a video encoding. The corresponding spectrogram of the noisy speech is encoded in a similar fashion into an audio encoding. A single shared embedding is obtained by concatenating the video and audio encodings, and is fed into 3 consecutive fully-connected layers. Finally, a spectrogram of the enhanced speech is decoded using an audio decoder.

Neural Network Architecture

The speech enhancement neural network model gets two inputs: (i) a sequence of video frames showing the mouth of the speaker; and (ii) a spectrogram of the noisy audio. The output is a spectrogram of the enhanced speech. The network layers are stacked in encoder-decoder fashion (Fig. 2). The encoder module consists of a dual tower Convolutional Neural Network which takes the video and audio inputs and encodes them into a shared embedding representing the audio-visual features. The decoder module consists of transposed convolutional layers and decodes the shared embedding into a spectrogram representing the enhanced speech. The entire model is trained end-to-end.

- **Video encoder**
  The input to the video encoder is a sequence of 6 consecutive gray scale video frames of size 128×128, cropped and centered on the mouth region.
  The video encoder has 6 consecutive convolution layers. Each layer is followed by Batch Normalization, Leaky-ReLU for non-linearity, max pooling, and Dropout of 0.25.

- **Audio encoder**
  Both input and output audio are represented by log mel-scale spectrograms having 80 frequency intervals between 0 to 8kHz and 20 temporal steps spanning 200 ms. The audio encoder is designed as a convolutional neural network using the spectrogram as input. The network consists of 5 convolution layers. Each layer is followed by Batch Normalization and Leaky-ReLU for nonlinearity. We use strided convolutions instead of max pooling in order to maintain temporal order.
- **Shared representation**
  The video encoder outputs a feature vector having 2,048 values, and the audio encoder outputs a feature vector of 3,200 values. The feature vectors are concatenated into a shared embedding representing the audio-visual features, having 5,248 values. The shared embedding is then fed into a block of 3 consecutive fully connected layers, of sizes 1,312, 1,312 and 3,200, respectively. The resulting vector is then fed into the audio decoder.
- **Audio decoder**
  The audio decoder consists of 5 transposed convolution layers, mirroring the layers of the audio encoder. The last layer is of the same size as the input spectrogram, representing the enhanced speech.
- **Optimization**
  The network is trained to minimize the mean square error (l_2) loss between the output spectrogram and the target speech spectrogram. We use Adam optimizer with an initial learning rate of $5e^{-4}$ for back propagation. Learning rate is decreased by 50% once learning stagnates, i.e. the validation error does not improve for 5 epochs.

Audio post-processing

The speech segments are inferred one by one and concatenated together to create the complete enhanced spectrogram. The waveform is then reconstructed by multiplying the mel-scale spectrogram by the pseudo-inverse of the mel-spaced filter bank, followed by applying the inverse STFT. We use the original phase as of the noisy input signal.

Real-Time Operating System

We implemented a new product model to real-time operating system by simulating an audio-video recording in real-time. The input data with noise is provided by the actual environment, while making a real-time prediction using an audio-visual end-to-end neural network model. Since our project is a real-time system, it was crucial for the system to be fast, efficient and clear. In order to optimize the calculations made by the system, we used parallel computation to speed the process time, and to prevent unnecessary calculations.

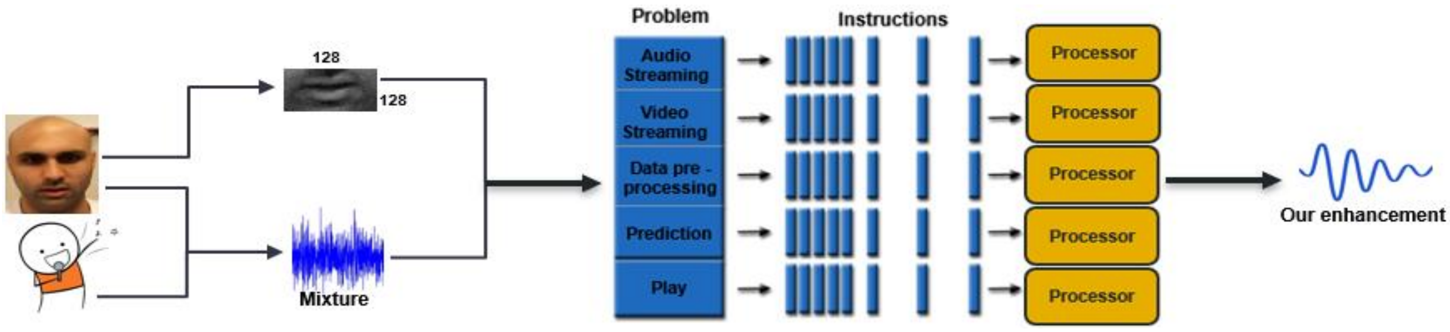


Figure 3: Illustration of our real-time operating system based on parallel computation. Every element is independent to each other, and therefore, the system is fast and efficient.

- **Audio-Video Streaming**
  We independently process the audio and video from the camera, with a sequence of every 6 frames being stored separately in the queue, and the corresponding voice being saved respectively in a different queue.
- **Date Pre-processing**
  When there is an element of 6 frames in the queue, and when the audio is in the second queue, we begin to process the information, i.e. identify and cut the lips, and accordingly create the appropriate spectrogram. They both enter as input to the neural network.
- **Prediction**
  The neural network performs the prediction according to the inputs it receives.
- **Play**
  The waveform is reconstructed from the spectrogram, and we play the voice in real time.

Materials, hardware and software packages

The developing of the system took place on a Python virtual environment - Linux OS, using an IDE dedicated to developing Python applications - Pycharm.

We used a computer with strong GPU build, and camera with built-in microphone for preparing the data and running the system.

Moreover, we used python libraries and development tools:

- SciPy / NumPy – A library used for scientific computing and technical computing.
- Dlib – A library to detect and extract facial landmarks from an image.
- OpenCV – A library of programming functions mainly aimed at computer-vision.
- TensorFlow / Keras – A library used for machine learning applications such as neural networks.
- Pyaudio - A library to play and record audio on a variety of platforms using python.

Implementation details

As described in the appendix.

Evaluation

Evaluation dataset and performance metrics

In order to evaluate our work, we use a dataset containing videos of Yakov Monfred speaking. This dataset consists of 260 videos, each of 1-2 minutes long. The dataset was recorded in quiet room and includes an unbounded vocabulary. We verified the robustness and performance of our model on several speech enhancement tasks (English, Hebrew and ambient noises) using our dataset.

In all cases, English background speech was sampled from the LibriSpeech [10] dataset. The Hebrew background speech was sampled from news videos covering different noise types. For the ambient noise we used different types of noise such as rain, motorcycle engine, basketball bouncing, etc. The speech and noise signals were mixed with SNR of 0 dB both for training and testing. In each sample, the target speech is mixed with background speech or ambient noise.

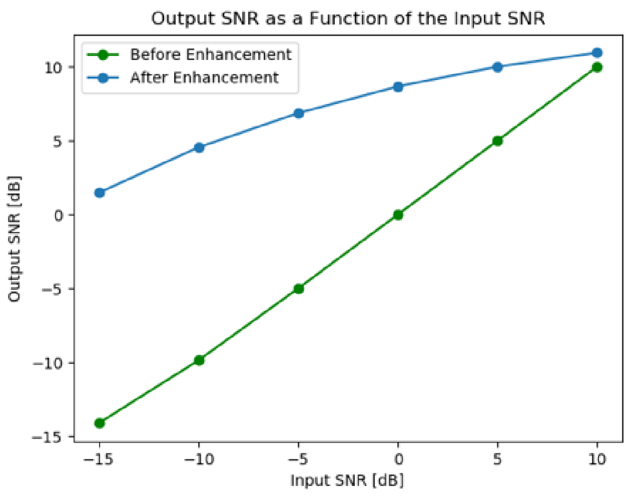
We reported an evaluation using three objective scores: SNR (Signal-to-noise ratio) for measuring the noise reduction, PESQ (Perceptual Evaluation of Speech Quality) for assessing the improvement in speech quality and Real-Time score (execution delay without data drifting). Since listening to audio samples is essential to understand the effectiveness of speech enhancement methods, supplementary material is available online[1].

Results:

We report an evaluation using three objective scores:

SNR (Signal-to-noise ratio)

	SNR (dB)					
Input SNR (dB):	-15	-10	-5	**0**	5	10
Before Enhancement	-14.10	-9.86	-4.99	**0**	5	10
After Enhancement	1.47	4.54	6.85	**8.66**	10	10.94

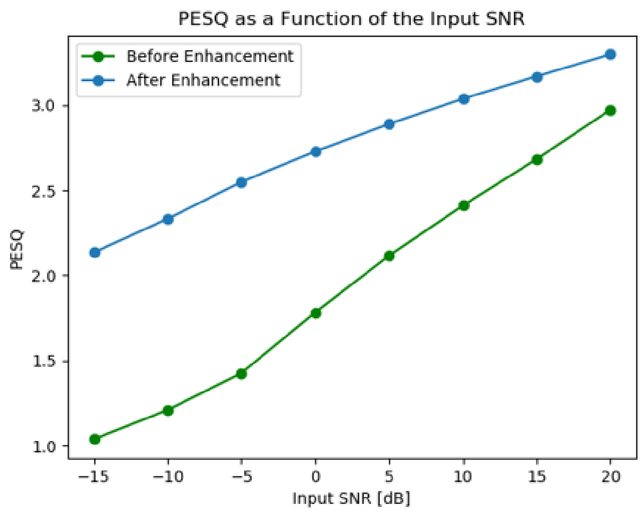


[1] Examples of speech enhancement can be found at https://drive.google.com/open?id=18-Hu5NfcZSO8dQ-mfoJ86mMyRBCTSWOH

For input SNR of 0 dB, we got SNR of 8.66 dB after enhancement.
We can notice the convergence around 10 dB, thus when the noise is very weak, our system is outputting a result very close to reality.

PESQ (Perceptual Evaluation of Speech Quality)

	PESQ							
Input SNR (dB):	-15	-10	-5	**0**	5	10	15	20
Before Enhancement	1.03	1.21	1.42	**1.78**	2.11	2.41	2.68	2.97
After Enhancement	2.13	2.33	2.54	**2.72**	2.88	3.03	3.16	3.29



For input SNR of 0 dB, we got enhanced PESQ of 2.72 compared to mixture PESQ (original sound with noise) of 1.78 before enhancement. We can notice again the convergence around 20 dB, thus when the noise is very weak, our system is outputting a result very close to reality.

Real-Time (execution delay without data drifting)

260 ms

Regarding real-time score for measuring end-to-end delay in real-time systems to determine if the system adds an execution delay. We got an execution delay of 260 ms without data drifting. The steps of calculation are as following: capture 6 frames take 200 ms.
In the remaining time - 60 ms, the system is working in parallel: preprocess, network prediction and real-time play.

We achieved significant improvement both in noise reduction and speech quality in the different noise types, and thus succeeded to improve the quality and intelligibility of noisy speech.

Conclusion

An end-to-end neural network model, separating the voice of a visible speaker from background noise and other speakers using visual information from the target speaker's lips, has been presented. This model builds a system that is robust to similar vocal characteristics of the target and noise speakers, and makes an effective use of the visual information.
The proposed model consistently improves the quality and intelligibility of noisy speech. Finally, we demonstrate for the first time, real-time audio-visual speech enhancement on a general dataset not designed for lipreading research that was created personally.
Our model is compact, and operates on short speech segments (200 ms), and thus suitable for real-time applications.

It is important to note that our proposed approach is speaker-dependent, thus a new model needs to be trained for each new speaker. Therefore essentially, the model was trained on Yakov Monfred. Achieving speaker-independent speech enhancement is a non-trivial task, and is out of the scope of this project.

Future Work

There are various possibilities that can be pursued and implemented, in different planes.

In order to elaborate the usage of our algorithm, we might consider achieving speaker independent speech enhancement by isolating a single speech signal from a mixture of sounds, such as other voices and background noise. The goal is to produce videos in which speech of specific people is enhanced while all other sounds are suppressed in real-time.

Another way to improve the usage of the algorithm is to implement a new app extension (Skype, WhatsApp, Viber). Build a plugin to enable real time video conference using mobile phone or computer, and perform speech enhancement with our neural network model.
The clean voice output will be played through the app.

Our UI/UX capabilities were very limited. In our opinion, there is still work left and a lot of options that could help benefit and enrich the UI an improve the application's user experience.

Bibliography

1. Y. M. Assael, B. Shillingford, S. Whiteson, and N. de Freitas, “Lipnet: Sentence-level lipreading,” arXiv:1611.01599, 2016.
2. X. Lu, Y. Tsao, S. Matsuda, and C. Hori, “Speech enhancement based on deep denoising autoencoder.”
3. S. Pascual, A. Bonafonte, and J. Serra, “Segan: Speech enhancement generative adversarial network,” arXiv:1703.09452, 2017.
4. Y. Isik, J. L. Roux, Z. Chen, S. Watanabe, and J. R. Hershey, “Single-channel multi-speaker separation using deep clustering,” in ICASSP’16, 2016.
5. F. Khan and B. Milner, “Speaker separation using visually-derived binary masks,” in Auditory-Visual Speech Processing (AVSP), 2013.
6. J.-C. Hou, S.-S. Wang, Y.-H. Lai, J.-C. Lin, Y. Tsao, H.-W. Chang, and H.-M. Wang, “Audio-visual speech enhancement based on multimodal deep convolutional neural network,” IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 2, no. 2, pp. 117–128, 2018.
7. A. Gabbay, A. Ephrat, T. Halperin, and S. Peleg, “Seeing through noise: Speaker separation and enhancement using visually-derived speech,” in ICASSP’18, 2018.
8. A. Ephrat, T. Halperin, and S. Peleg, “Improved speech reconstruction from silent video,” in ICCV’17 Workshop on Computer Vision for Audio-Visual Media, 2017.
9. Aviv Gabbay, Asaph Shamir and Shmuel Peleg. “Visual Speech Enhancement”, The Hebrew University of Jerusalem, 2018.
10. V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “Librispeech: an asr corpus based on public domain audio books,” in ICASSP’15, 2015, pp. 5206–5210.

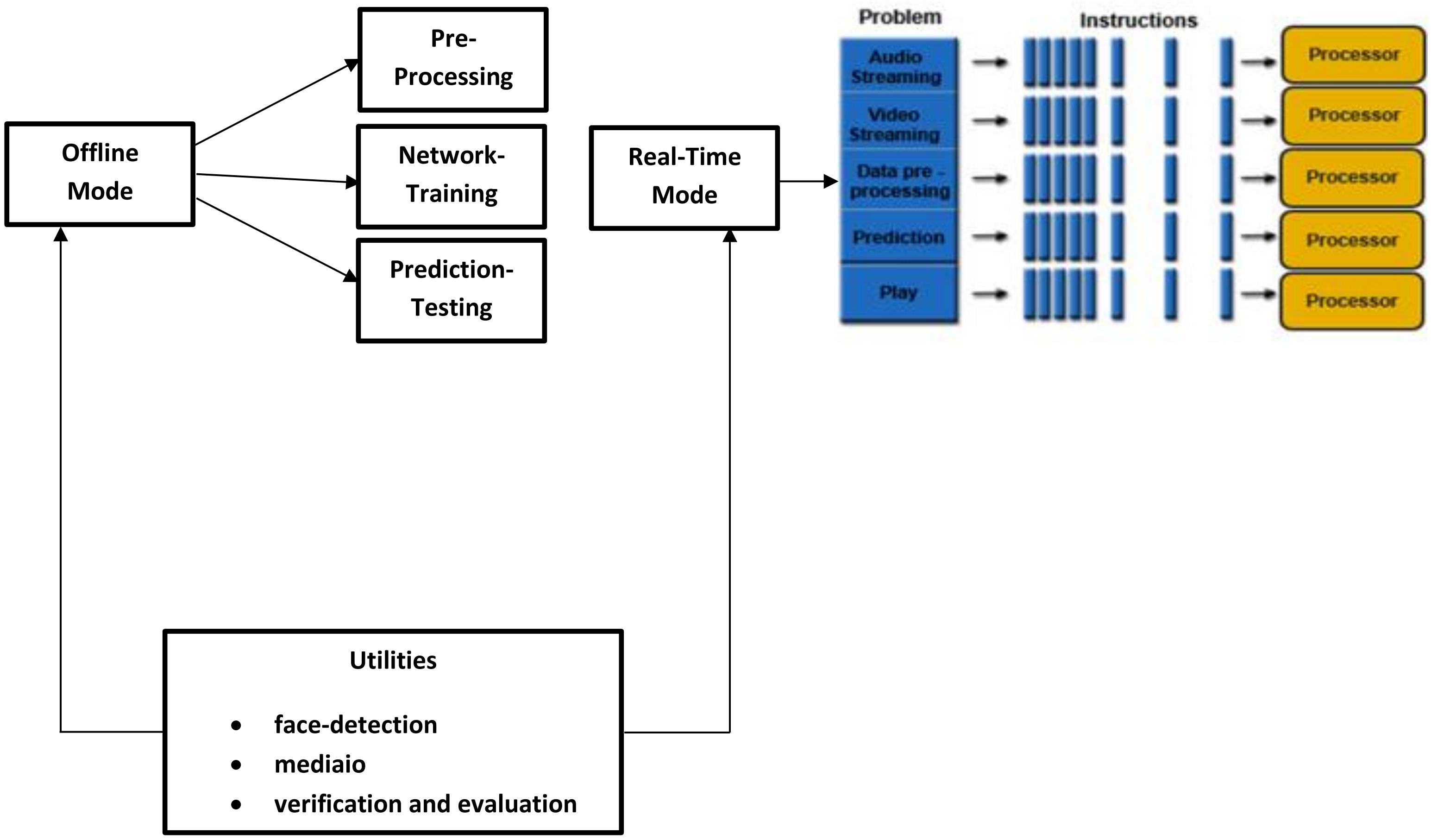
Appendix

Implementation details

Our project consists of three main parts. The first part is the pre-processing stage where we process the audio and video in as small segments as possible in order to the system to work in real time without any delays. The second part consists of training the neural network on the processed information. Finally, in the third part we make predictions, and in this way, we filter out non-target background noise.

The project was built in two stages. In the first stage, we developed and ran everything in offline mode and saw that we did reach the desired results according to the metrics.
We then developed the system to work in real time in parallel (second stage).

Structure of components



Classes & Files description

Below is a full description of all project files and their role in the system.

Pre-Processing and Neural Network Files

data_processor.py

- Contains two main methods, preprocess_video_sample and preprocess_audio_signal, that are responsible for the audio and video pre-processing.
- Other methods included in this file are responsible for the reconstruction of the waveform from the mel-scale spectrogram (audio post-processing).

dataset.py

- This file contains two classes that are responsible for processing audio and video data files from the computer in offline mode.

network.py

- This file contains all the neural network architecture as explained above (see algorithm details part).

speech_enhancer.py

- The main file responsible for running and performing the following actions according to the inputs: pre-processing, training and testing (offline mode).

face-detection

- A python wrapper for detecting face and mouth regions in images using dlib library.
- In addition, there is a file called "shape_predictor_68_face_landmarks.dat" that is used to find features of interest inside the video for mouth-cropping.

mediaio

- A python wrapper for reading, writing and manipulating audio and video files.

Verification and Evaluation Files

speech_enhancement_evaluator.py

- A python program that allows us to calculate the PESQ score.
- pesq – an executable program.

snr.py

- A python program that allows us to calculate the SNR score.

Real-Time Files

real_time_speech_enhancer.py

- The main file that runs the whole system in real time. In this file all work is done in parallel: audio streaming, video streaming, data pre-processing, prediction and output play. It is possible to save the results of the run on the computer, and show the total execution time.

real_time_network.py

- This file contains all the neural network architecture as explained above (see algorithm details part) with real-time code adjustments. For the Keras library to work in real time in parallel, and for the neural network to work optimally, we needed to make real-time adjustments.

real_time_audio_io.py and real_time_video_io.py

- A python wrapper for reading, writing and manipulating audio and video files in real time.

real_time_face_detection.py

- A python wrapper for detecting face and mouth regions in images using dlib library in real time.