CS 419 Project Presentation

Solving the cocktail party problem using Deep Clustering

Aditya Sriram
Jujhaar Singh
Raavi Gupta
Rishabh Ravi
Vedang Gupta

Outline

Introduction

- Deep clustering
- Problem statement

Dataset

- Exploring LibriMix

Algorithm

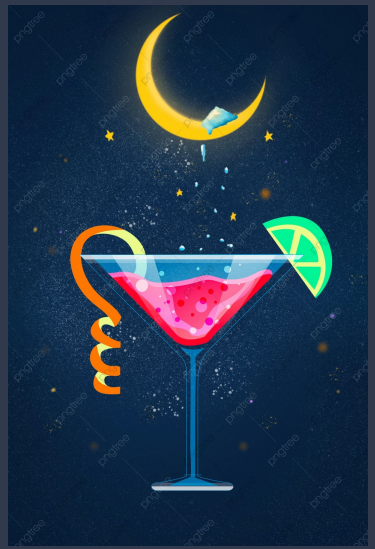
- Extracting data
- Data pre processing
- Building the neural network

Conclusion

- Our results
- Future work

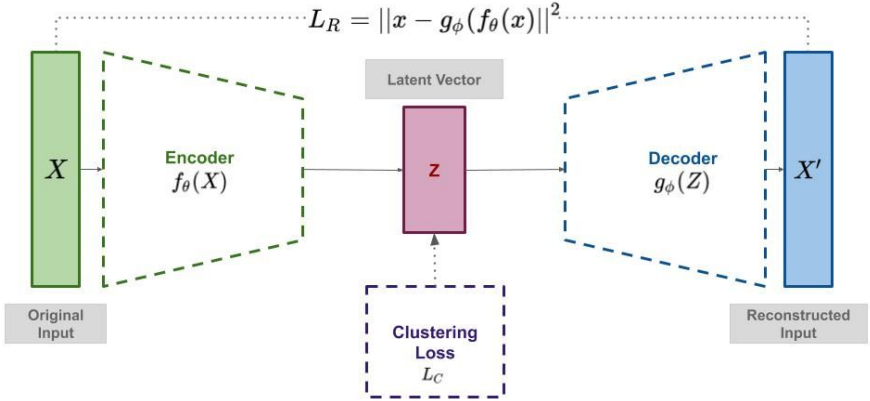
Introduction

We address the problem of ”cocktail-party” source separation in a deep learning framework called deep clustering.



Deep clustering

- Deep clustering frameworks combine feature extraction, dimensionality reduction and clustering into an end to end model, allowing the deep neural networks to learn suitable representations to adapt to the assumptions and criteria of the clustering module that is used in the model.



Introduction - Problem Statement

The “**cocktail party problem**” is encountered when sounds from different sources in the world mix in the air before arriving at the ear, requiring the brain to estimate individual sources from the received mixture.

The **cocktail party effect** is the phenomenon of the brain's ability to focus one's auditory attention on a particular stimulus while filtering out a range of other stimuli, such as when a partygoer can focus on a single conversation in a noisy room.

We seek to artificially replicate the cocktail party effect - of separating certain audio sources from the rest using machine learning algorithms.

The Dataset

- We used an open source dataset ‘LibriMix’ which is used for speech separation in noisy environments.
- It mixes signals from LibriSpeech (a clean dataset) and mixes them with WHAM noises.
- This dataset allows us to select the number of speakers as well as whether noise should be incorporated or not.
- For this project we have used a mixture of 2 speakers without WHAM noise.

The Algorithm

- Creating the dataset
  - We first sample the .wav files at 8kHz and then takes its STFT(short time Fourier Transform), which was fed as input.
- Building the target Y
  - The binary masks were used to build the target Y to train our network.
  - In each time-frequency bin, the mask values are set to 1 for the source with the maximum magnitude and 0 for the other.
  - To avoid training the network to assign embeddings to silence regions, a binary weight for each time-frequency bin was used during the training process, only retaining those bins such that magnitude of the mixture at that bin is greater than some ratio of the maximum magnitude.

The Algorithm

- Making the model
  - A deep network then assigns embedding vectors to each time-frequency region of the spectrogram, according to an objective function that minimizes the distances between embeddings of time-frequency bins dominated by the same source, while maximizing the distances between embeddings for those dominated by different sources.
  - The network structure used in our experiments has two bi-directional long short-term memory (BLSTM) layers, followed by one feedforward layer. Each BLSTM layer has 600 hidden cells, with Tanh as the activation.
  - In each updating step, to avoid local optima, Gaussian noise with zero mean and 0.6 variance was added to the weight.

The Algorithm

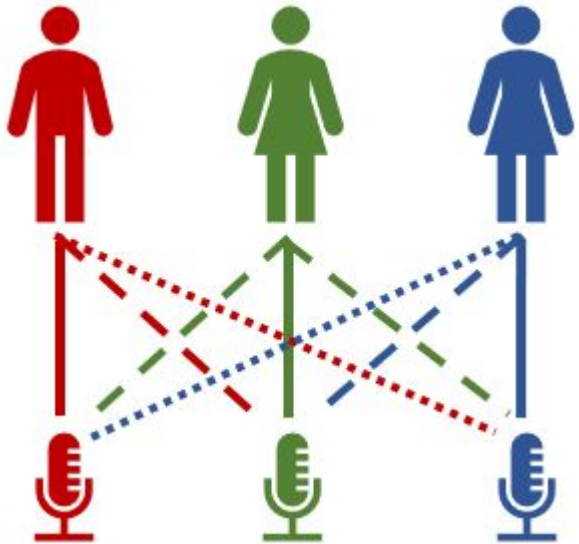
- Defining the Loss
  - The clustering error between the estimates $\bar{Y}$, and the labels $\mathring{Y}$ were given as

$$\|\bar{Y}(\bar{Y}^T\bar{Y})^{-1}\bar{Y}^T - \mathring{Y}(\mathring{Y}^T\mathring{Y})^{-1}\mathring{Y}^T\|_F^2$$

- Selecting an Optimizer
  - The model was optimized using Stochastic Gradient Descent with momentum 0.9 and a fixed learning rate of 0.00001.

Future work

- With our success in separating two audio sources, we look forward to extend the same results with multiple audio sources.
- We could also incorporate noise into our audio.
- Look for methods to speeden the model.



Thank You!