Voice-Enhancement-using-Deep-Learning

While dealing with real time audio speech signals, we are always concerned about the various types of noises that are added to the original signal which results in corruption of the original clean signal. In order to retrieve a better-quality signal, it is essential to enhance these signals by eliminating the noises present in them. Here this process is attempted to achieve through Deep Learning.

Here I have attempted to design a DL pipeline for audio enhancement. This is basically a DNN based filter that takes real world audio as input and removes noises to produce a clean audio.

Dataset:

The dataset used for this project is a curated dataset taken from NOIZEUS speech corpus, a noisy speech corpus. The noisy database contains 30 IEEE sentences (produced by three male and three female speakers) corrupted by eight different real-world noises at different SNRs. The noise was taken from the AURORA database and includes suburban train noise, babble, car, exhibition hall, restaurant, street, airport and train-station noise. Here the sentences were originally sampled at 25 kHz and down-sampled to 8 kHz.

Used dataset: https://drive.google.com/drive/folders/1y-_HbXrJMKo0QXlTRNTmGOuZU1ntsW2i?usp=sharing

Downloadable link for the dataset:

You can find the complete dataset here https://ecs.utdallas.edu/loizou/speech/noizeus/

Approach:

- Input is taken as audio **.wav** format.
- **Clean audio** and **audio with airport noise** are used for training and **audio with babble noise** used for testing.
- The data is loaded, **STFT** is performed and **training data is normalized** using a function **loadfile(…).**
- A fully connected **DNN** with **5** hidden layers with **1000** hidden units each for speech denoising.
- **tanh** activation functions are used for the hidden layers.
- **Xavier initialization** has been used for weights along with **batch normalization** and **Adam Optimizer**.
- **relu** activation function is used for the last layer, due to **non-negative output requirement** and to. **avoid vanishing gradient problem.**
- The network is trained for **700 Epochs** with a **learning rate of 0.0001**.

- **Mini-batch learning** is applied with a **batch size of 4, using Random shuffling of the training data.**
- **30% dropouts** are applied during training to **avoid overfitting**.
- Model tested and **clean audio reconstructed as .wav files**.

Thank You...