U-Net Noise Removal

Member & Workloads

工科海洋系大四 張在然 b06505002
Data collection/preprocessing, report
工科海洋系大四 陳奕舟 b06505006
Implementation of U-Net structure, report

Introduction

To achieve noise robustness, many different approaches were introduced in chapter 15. In our project, we used a learning based method called U-Net to perform noise removal of the sound sample.

Dataset - https://github.com/microsoft/MS-SNSD

This dataset contains a large collection of clean speech files and variety of environmental noise files. We randomly picked the files in there (5000+ clean speech sample plus 5000+ noise sample) and combined them into our training data.

Implementation

Step 1. Add noise to clean speech to generate input data

In order to train a Unet model, we need the inputs:

- Clean speech samples mixed with noise (X_{mix})
- Original clean sample (ground truth y_{clean})

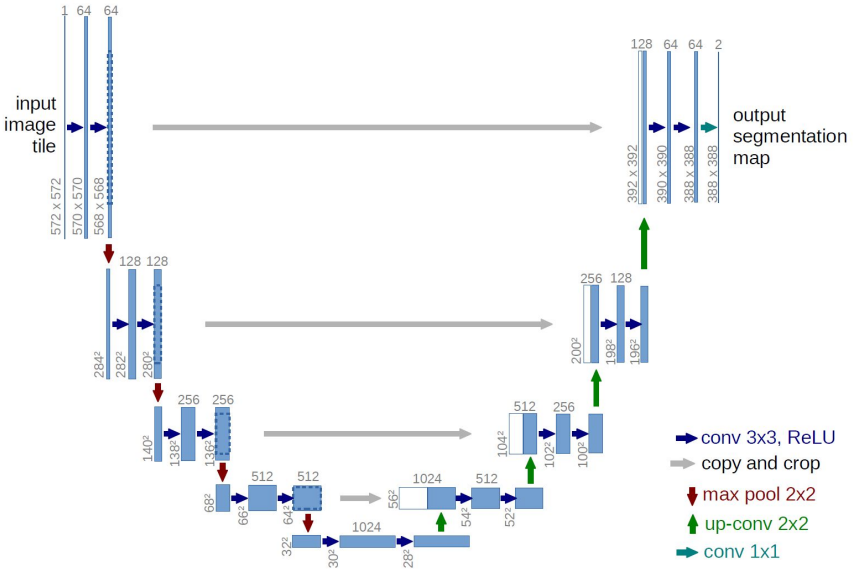
We loaded the required files with Librosa as y_{clean} and y_{noise} , with a sampling rate of 16000, and extracted the first 33000 sample points, which was about 2 seconds.

Then, we combined y_{clean} and y_{noise} to acquire mixed sample X_{mix} as the training data.

Step 2. Convert sound wave into spectrogram

Next, both y_{noise} and X_{mix} were converted into spectrograms using short-time Fourier transform, with the number of fft set to 512. This gave us an output spectrogram of size (257,257), which was then saved into a h5py file. We prepared 5000 spectrogram sets as the training data, and 1000 more for validation.

Step 3. Unet



This is the structure of our neural network. Since we have converted the sound files into spectrograms, we can feed them into the network as images.

The Unet was implemented with a series of down-sampling and up-sampling. During the down-sampling process, the information in the hidden layers were reserved as a reference in the up-sampling process. This was an improvement over the previous Auto Encoder/Decoder structure.

The output of the network was going to be an image with the same size as the input. To have the network learn over the training process, we defined the training loss to be the pixel-wise difference between the model output and the ground truth (y_{clean}). The network will gradually try to lower the difference between them through each iteration.

Step 4. Training

In order to solve the problem that our input data, as the result of short-time Fourier transform, had contained imaginary numbers, we decided to train two seperate networks with the same structure, one for the real part (magnitude), and the other for the imaginary part (phase).

For the training variables, we set the batch size to be 64, with a learning rate of 0.001, optimized with Adam optimizer, over 75 training epochs. The training loss was defined as the mean-squared error (nn.MSELoss) between the output and the ground truth.

To save memory consumption, on the upsampling part of the network, we used nn.ConvTranspose2d instead of nn.Upsample, since the latter involved the slower bilinear transformation.

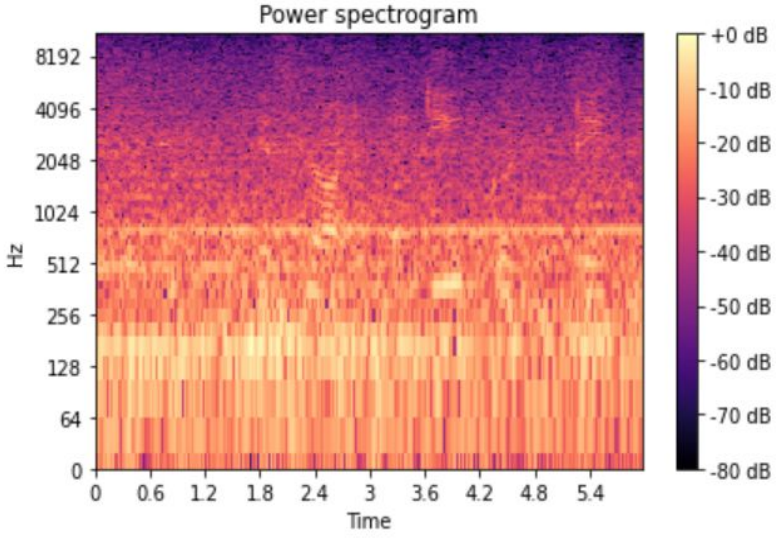
Step 5. Prediction and inverse transformation

After about 4 hours for each model, we can finally try generating output with them. Again, the testing data (which was taken from files unseen by the model during training) was converted to spectrogram, and separated into real and imaginary parts.

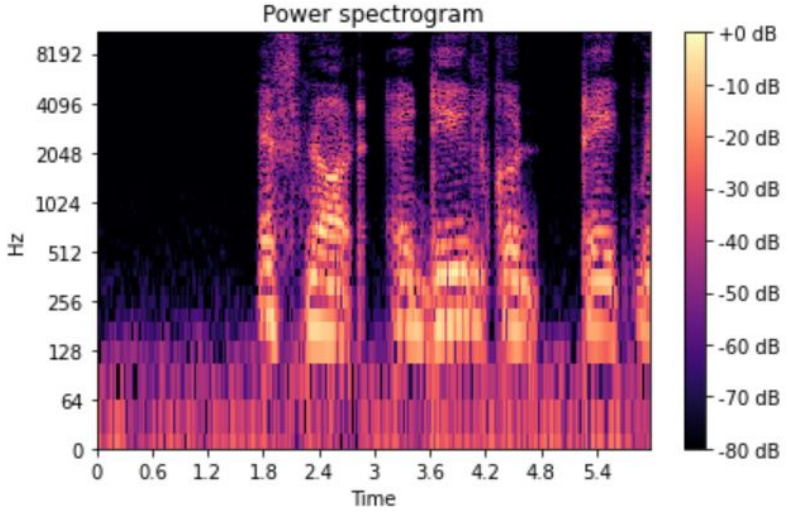
Finally, we took the output of both models' predictions, and combined them together into one final spectrogram. With the help of Librosa's inverse short-time Fourier transform function, we can get the final result y_{pred}.

Result

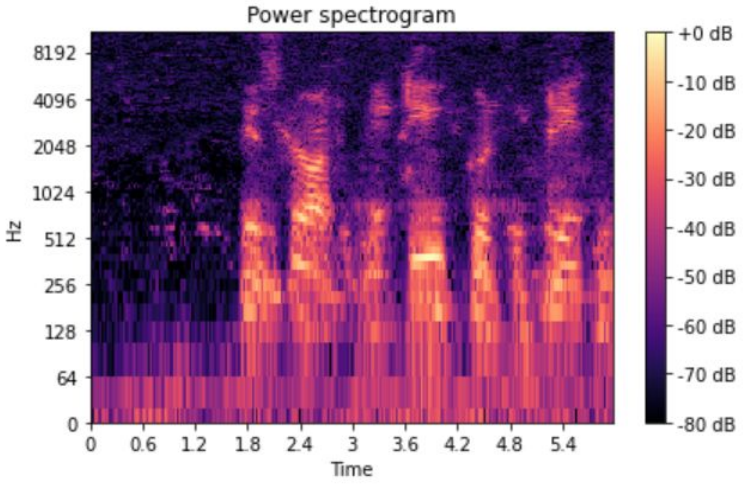
Input (mixed with noise)



Ground truth (clean speech)



Output (prediction)



To see (listen) the result of noise removal (.wav file):
https://drive.google.com/drive/folders/1jwJQqUw3j7RjvBA2m_qixM9UXFQ5F4Rr?usp=sharing

Conclusion

From the output we obtained, we can see the model successfully separated out the desired result, with noise almost removed. However, over the past years, there have been many new neural network structures that improved over the performance and efficiency of this particular task. Also, from the information we obtained, we figured that there should've been some work-around to take the entire spectrogram as an input for one single model, instead of separating it into real and imaginary parts like we did. These are the main topics we can further improve upon.

Reference

U-Net:

https://github.com/milesial/Pytorch-UNet

Stft & istft:

http://man.hubwiz.com/docset/LibROSA.docset/Contents/Resources/Documents/generated/librosa.core.stft.html

Wave U-net:

https://arxiv.org/abs/1806.03185

Our code and some prediction result:

https://drive.google.com/drive/folders/1jwJQqUw3j7RjvBA2m_qixM9UXFQ5F4Rr?usp=sharing