Audio Super-Resolution

Şut George-Mihai
3rd-year undergraduate, Computer Science
Babeş-Bolyai University
georgesut@yahoo.com

***Abstract*—Audio super-resolution refers to the task of increasing the sampling rate of an audio signal by training a neural net to produce outputs whose sampling rate is higher by a specific factor (x2, x4, x6 etc.).**

I. Introduction

In this paper, the goal is to investigate on whether a neural net can be trained with low-resolution audio data given as an input to produce super-resolution audio (i.e a reconstructed high-resolution audio signal). The point of the model is to predict the samples which are missing from the audio signal, which in this case will consist of pairs of high-res and low-res samples of sound clips containing vocal recordings from the VCTK dataset. The project has been inspired by image super-resolution and especially by time-series super-resolution. (Kuleshov, Enam, and Ermon [2], Hetherly [1])

The process of audio super-resolution using neural nets (also called bandwidth extension) is explained in Kuleshov, Enam, and Ermon [2] which states that the goal is to reconstruct a low-resolution signal with a sample rate R_1 into a high-resolution signal with a greater sample rate R_2. The paper clarifies the concept by giving a simple example of a 4 KHz signal being upsampled through audio super-resolution to a 16 KHz signal by a factor of 4. The audio signal is encoded into a spectrogram which displays the frequencies contained in the signal and the sound intensity in decibels.

A bottleneck-type architecture has been used, similar to the U-Net architecture, containing residual connections between pairs of layer b and layer $B - b + 1$, where B is the number of layers in the network. The first part of the network is responsible for downsampling data, whereas the second part upsamples it.

The model is trained on data containing high-resolution audio clips mapped to their low-resolution counterparts obtained by downsampling clips from VCTK, a popular speech dataset, and the PIANO dataset.

There are essentially two reference points to which the problem solved by Kuleshov, Enam, and Ermon [2] is compared: cubic spline interpolation and the dense neural network described in Li et al. [3], which targets the prediction of the phase and magnitude of the high frequencies in the signal. The loss function used is the mean-squared error, computing the sum of the squared differences between the low- and high-resolution signals, while the main metric that is highlighted is the signal-to-noise SNR ratio, often used in the signal processing domain.

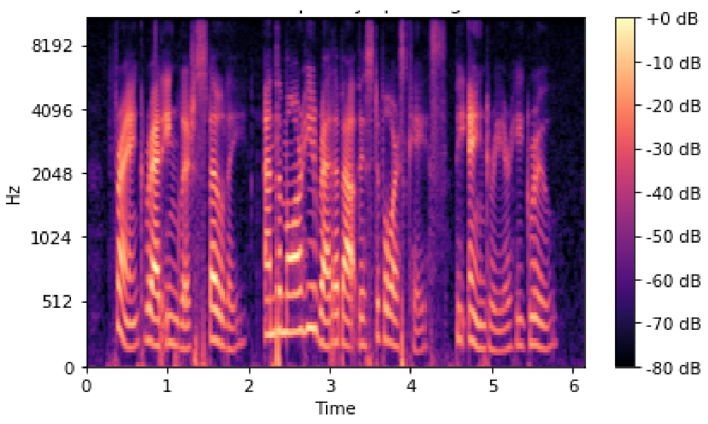

Fig. 1. Example of a spectrogram where time is shown on the x axis, the frequencies out of which the signal is composed are shown on the y axis and their corresponding magnitudes in decibels are displayed in the colorbar on the right.

The evaluation results in the paper show that the model outperforms the other referenced tasks, a fact which is also underlined by the MUSHRA test of individuals' ratings. The experiment conducted on the MagnaTagATune dataset in Kuleshov, Enam, and Ermon [2] shows that applying this architecture on music leads to poor results which could be improved with a larger and more computationally demanding model, so the model is mostly suitable only for vocal recordings, meaning that it can be useful in voice-over-IP applications.

Concluding the study, some of the impediments of the model are the lack of diverse data and the requirement for solid computing power, leading to results for a music dataset that are weaker than the cubic spline interpolation baseline.

The second article (Hetherly [1]) mentions that the presented implementation has only been trained on 10 epochs, hence the mediocre outputs, compared to the 400-epoch model described by Kuleshov, Enam, and Ermon [2]

II. Implementation

The loss function used is the mean-squared error, while the metrics are the signal-to-noise ratio and the normalised root mean-squared error (which doesn't highlight any important characteristics about the result). The optimizer is Adam. Another important detail is the number of residual blocks used in the model, which is 6.

The network has been trained in two ways:

1) 100 epochs, with batches of size 16
2) 1000 epochs, with batches of size 16
3) 100 epochs, with batches of size 16 on all of the data, with a 90/5/5 split

III. Results

Compared to each of the methods, the third one seems to achieve the best performance so far, based on the MSE and SNR output (25940.2207, 22.95). The plots containing the evolution of the loss values during training and validation are located in the current directory.

IV. Conclusion

The network needs a lot of improvement, so the next steps would be to use more training epochs and more generated data, while also refining hyperparameters. Afterwards, a spectrogram-based solution could possibly surpass the performance of the current model.

References

[1] Jeffrey Hetherly. *Using Deep Learning to Reconstruct High-Resolution Audio*. 2017. URL: https://blog.insightdatascience.com/using-deep-learning-to-reconstruct-high-resolution-audio-29deee8b7ccd.

[2] Volodymyr Kuleshov, S. Zayd Enam, and Stefano Ermon. *Audio Super Resolution using Neural Networks*. 2017. arXiv: 1708.00853 `[cs.SD]`.

[3] Kehuang Li et al. "Dnn-based speech bandwidth expansion and its application to adding high-frequency missing features for automatic speech recognition of narrowband speech". In: (2015).