Audio Super Resolution

Akarsha Sehwag 2015010
Manik Arora 2015053
Shiven Mian 2015094
YS Ramya 2015117

1. Introduction

1.1. Problem Statement and Motivation

Given an audio signal S1 at a low resolution, the task is to generate a higher resolution version signal S2 of S1, where the sampling rate of S2 ¿ S1, using different ML models. This is similar to Image Super Resolution, where the task is to super-resolve the image at high upscaling factors.

Audio Super Resolution has practical applications in telephony, compression, and text-to-speech generation; and neither of us has worked with signals before, giving us a motivation to pick up this project, also whatsapps poor quality audio was a big motivator for us.

2. Related Work

Audio Super resolution using CNNs: https://arxiv.org/pdf/1708.00853.pdf
Image super resolution using GANs: https://arxiv.org/abs/1609.04802

R1-¿ VCTK Dataset: http://homepages.inf.ed.ac.uk/jyamagis/page3/page58/page58.html
R2-¿ Piano Dataset: https://archive.org/search.php?query=piano%20beethoven

There have been two known previous works in this problem. Kuleshov et al (2017) use a deep learning based approach (a deep ResNet) for Audio Super resolution. Another previous work (Dong et al (2015)) use a Dictionary Learning based approach. We aim to simulate the results of Kuleshov et al, along with trying a GAN based model and some simpler deep models.

We are using two datasets - VCTK Corpus[R1] and a Piano Dataset[R2]. The VCTK Dataset is a set of wav files, with 109 native speakers speaking 400 sentences each. All the audio has been recorded in identical conditions. And the Piano dataset contains publicly available Beethovans Sonatas, consisting of around 10 hours of music.

3. Dataset and Evaluation

3.1. Pre processing and Data Division

The wav files were sampled at required sampling rate (16000 in our case), and the values were stored as a float array. This was further used in creating its low resolution counterpart where we randomly removed the data after regular intervals and replaced them by zero values. The zero values were then interpolated from the resultant array to give the final float array representation of low resolution data. Hence, we have pairs of low resolution data (input) and high resolution data (true value output) for all the audio files. We will split the dataset as 88%-6%-6%. Therefore, for VCTK dataset, around 9 speakers will be used for testing and for Piano Dataset, out of 32 publicly available Beethovans Sonatas, we'll use 1 for testing.

3.2. Evaluation Metrics

The performance of the model can be evaluated using the following two features of the input audio sample, actual audio sample and generated super resolved sample

1. **SNR**: Signal to Noise ratio

Given a signal y, and an approximation of the same signal x, the Signal to Noise ratio is given by:

$$\mathbf{SNR}(x, y) = 10 \log \frac{||y||_2^2}{||x - y||_2^2}.$$

Figure 1. Figure 2

Higher the SNR values, the clearer the sound

2. **LSR**: Log Spectral Distance

It's a distance measure to evaluate reconstruction quality of signals. It is given by: Where X and Xâre log-spectral

$$\mathrm{LSD}(x, y) = \frac{1}{L} \sum_{\ell=1}^{L} \sqrt{\frac{1}{K} \sum_{k=1}^{K} \left(X(\ell, k) - \hat{X}(\ell, k) \right)^2},$$

Figure 2. Figure 2

power magnitudes of y and x respectively. Lower the log spectral distance indicates matching frequency.

4. Analysis

Model 1: Auto-encoder

We used a standard model of three-layer Auto-Encoder to enhance the resolution of the audio file by reconstructing the low-resolution file. The Auto-Encoder model was set up as a Deep Neural Network with nodes in layers decreasing in the first half of the net, and then increasing successively towards the output layer, with exactly same number of nodes in the output layer as the number of datapoints in the high resolution audio file. The ground truth is the high-resolution file from which errors are evaluated.

Model 2: Convolutional Neural Network

This model and architecture has been implemented from the 2017 paper of Kuleshov et al. The model consists of downsampling blocks followed by upsampling blocks. One block contains a convolutional layer, a dropout and last a layer that adds non linearity. The non linearity layer was chosen to be Leaky ReLu and the rate of the dropout was put as 0.5. At each downsampling block, the spatial dimensions are halved and the filter size is doubled and the same is carried out in reverse during upsampling. This architecture, called the bottleneck was inspired by an auto encoder. During upscaling, in order to increase the time of the piece, a one dimensional version of SubPixel layer of Shi et al, 16 was implemented. The non-linearity layer was swapped with normal ReLU, and we also tried different alphas for leaky ReLU when not active and found 0.2 to be better than the rest. The dropout rate was fixed at standard 0.5 after trying 0.7 and 0.3 too and not finding it to affect the output much.

After trying out the different learning rates on a sample data file of a single speaker, the learning rate of 0.0003 was chosen. With a lesser learning rate it was taking too long to converge and with a higher learning rate, the accuracy was stagnant after a certain point.

Model 3: Generative Adversarial Networks

Super-resolution Generative Adversarial Networks have been used for image super resolution tasks till now. The framework works by taking in positive and negative samples in a generator and discriminator network and the generator improves by trying to contest the discriminator network. We tried to modify an implementation of SRGAN to work for audio samples.

Model 4: Long Short Term Memory

Long Short Term Memory networks are a special kind of RNN that is specifically used for sequence model. The intuition behind using an LSTM was because of its capability to generate from data that it has seen previously. So we train LSTM on both high-res and low-res signals so that it learns how to model the high-res signals and fill in the missing values. Since audio being a sequence model, it can be used to train an LSTM.

Model 5: Hidden Markov Model

The intuition behind using HMM came from the fact that it can be used for audio generation problems like Bandwidth extension, where we are supposed to reconstruct the high resolution audio from a low quality input which has a small fraction of the initial samples we considered. In Audio super-resolution also, we are reconstructing the audio by predicting the missing values of the low-resolution audio. This is done similar to image super-resolution, since we are training our model on pairs of low and high-resolution data. And while testing the data, we are predicting the value of the final audio signal based upon the trained data. The results produced from this are not really great.

4.1. Challenges Faced

1. **Technical Challenges:** Library issues in HPC, therefore we have currently used partial set of the entire data. (2432 audio samples of 3 seconds)
2. Scaling and Shifts in Results
3. Accuracy and performance of the trained models.

4.2. Design Choices

Explanation for Hyperparameters Choice:

Speed- Although Speed is not the most important factor when working with high capability processors, for initial testing on local machine, speed is a limiting factor. For speed purposes, ReLU worked faster than Sigmoid, as is apparent, since Sigmoid uses exponential function.

Accuracy - We obtained more accurate results by using ReLU Activations in both layers as compared to other combinations of Sigmoid and ReLU.

Convergence - We used smaller batch sizes which helped us reach stable kernel values faster as compared to larger batch sizes.

4.3. Supporting Evidence

According to Fig1 and Fig2 which are float arrays of the low resolution and high resolution graphs plotted respectively, it is apparent that values can be predicted between data points of the low resolution signal to add additional data points to transform it into a higher resolution signal. Hence, generative models are appropriate to predict new values between data points. Autoencoders can be used for the same purpose by trying to reproduce the high resolution signal via the neural net from the input of lower resolution signal.

Our current trained model could be improved to achieve better accuracy. We need to explore more hyperparameters and analyse their effect on training. Our model seems not to be overfitting or underfitting the data.

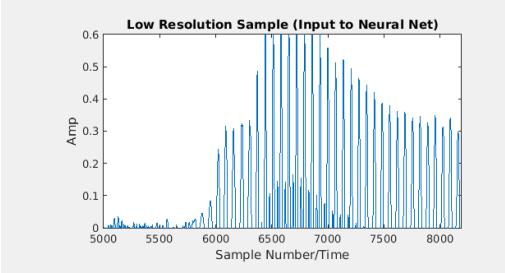

Figure 3.

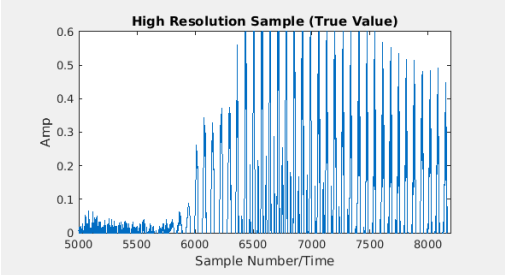

Figure 4.

4.4. Data Domain Analysis

Audio data has many characteristics which are reflected in the plots. The data points become more dense near every crest and trough and the sample appears to have continuous behaviour since audio is a continuous signal, and not a discrete one. Since we have discretized this continuous signal, we obtain values with frequent alternating maxima and minima. Our predicted values, if correct, should remain within the immediate maxima and minima (or exceed very slightly), otherwise we can end up with irrelevant audio data points.

4.5. Results

We have obtained newly generated files of better resolution than the input. Fig 4 shows the newly obtained waveform of a testing data for instance. We have also calculated the 2 evaluation metrics for our current model as shown in Fig 5.

As can be seen, our reconstructed signal from the Autoencoder isnt a big improvement over the low resolution signal, which was what we expected. We expect the generative models to give significantly better results. In case of Hidden Markov Model, the results for HMM shows us that it is definitely an improvement from the low-resolution signal since our Signal to Noise Ratio decreases significantly which shows that the sound quality improves.

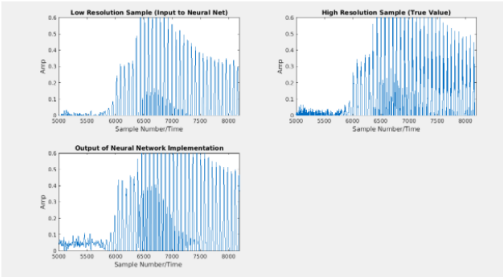

Figure 5. Autoencoder plot

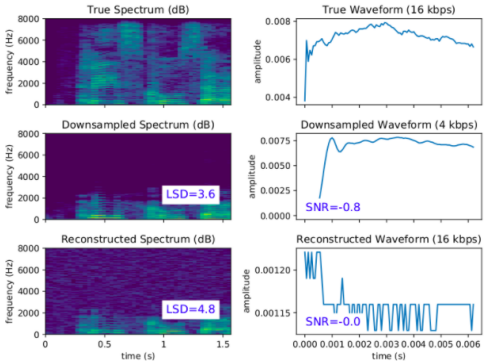

Figure 6. AutoEncoder Results

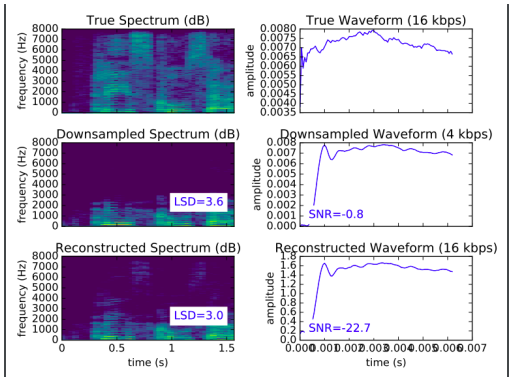

Figure 7. HMM Results

4.6. Interpretation of Results

The current results are satisfactory according to observation of the the generated waveforms which are similar to the high resolution audio waveforms visually. The quantitative measures show an increase in SNR as output of our model, which implies we have clearer audio, and increase in LSD which shows that our predicted values do not agree with the float values of our high resolution audio accurately, and improvement in model is required. Therefore we used Hidden Markov Models which showed better results as compared to that of the Auto-encoder. And then training it with even

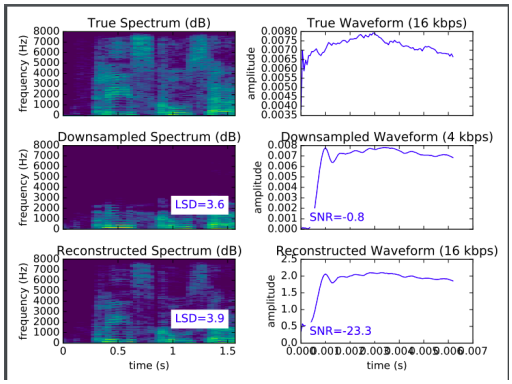

Figure 8. CNN Results

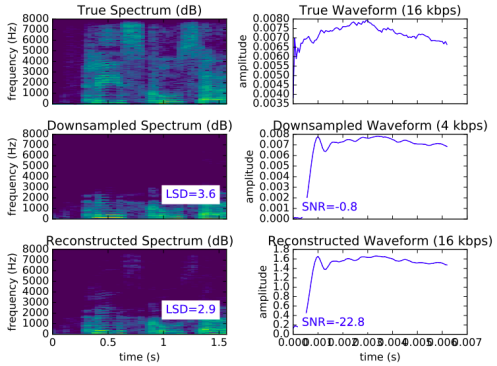

Figure 9. LSTM Results

better models like GANs and CNNs gave us almost perfect results.

4.7. Insights from Analysis

Our current tests show that our next immediate aim with the current model is to predict values closer to the true values of the high resolution audio. We do this by exploring more architectures and increasing the input values of our grid search. We have got clear audio in the current model which is, in fact, used as an input to the neural network which successfully increases LSD. Hence, via a hybridization of the current and the model we aim to obtain, we achieved a decrease in LSD and increase in SNR value in case of CNN and GANs. HMMs' are midway between the Auto-encoders and the CNN models.

4.8. Modification in project Proposal

We have selected the VCTK Corpus from the set of datasets provided in the proposal, and have included a new Piano Dataset as mentioned above. Also, instead of using a WGAN and VAE, we decided to try simpler models (LSTM and HMM) instead. The evaluation metrics remain the same as the proposal.

4.9. Contributions

Shiven Mian: Data Preprocessing, LSTM and SRGAN for Audio Resolution

Manik Arora: Data Preprocessing, Auto-Encoder and Hidden Markov Model,

YS Ramya: Data Preprocessing, Convolutional Neural Networks [Kuleshov et al. model (2017)],

Akarsha Sehwag: Data Preprocessing, Hidden Markov Models and CNN (Kuleshov et al).