EECS 545 Project Final Report: Audio Upscaling with Deep Learning

Adam Ingerman, Paul Knudsen, Vishnu Rengaraj, Eric Ward
{adaming, pknudsen, vrenga, ericward}@umich.edu

Abstract

With the continued growth of videoconferencing it is imperative that sound quality is maintained regardless of microphone quality. This work focuses on upscaling low resolution audio using deep learning to solve this problem. We have successfully implemented a novel frequency residual neural network that increases the frequency range of low sample-rate audio clips as well as tested the effect of generative adversarial networks and other low quality data forms on the upscaling process.

1 Introduction

In the current remote environment, almost all meetings take place in a virtual setting, either through Zoom or another video conferencing provider. However, there is a wide disparity between the devices that people use in terms of camera and audio quality. In particular, the quality of microphones widely ranges. Not everyone has the ability to upgrade their setup with high quality headphones and have an external microphone instead of the laptop built in microphone. Our neural network aims to mediate the hardware disparity by upscaling the audio to simulate a high quality microphone.

This issue may be less prevalent in business meetings, where laptops are standardized across entire companies and are designed for a work environment. However, for remote schooling, laptops are dependent on the student and the family, specifically on their financial status. By implementing audio up-scaling through the video conferencing provider, students will be able to better converse with their teachers. This application can be extended to anyone with a low quality microphone. They will be able to have a similar sound quality as someone with a higher-end microphone.

By solving this problem, low quality audio devices will be able to output sound similar to that of higher end devices. The current remote environment will not last permanently. However, certain remote aspects will stay. Video calls will still be a norm for a lot of business meetings and could expand school access to anyone with a laptop and internet connection. The audio upscaling would allow for a more consistent experience in all these meetings and lessons. It would be the first step in removing financial barriers from video conferencing and remote learning.

2 Background

This report deals with topics in audio engineering. A few important terms in this domain such as waveform, spectrogram, short-time Fourier transform, Peak Signal-to-Noise ratio, and Log-Spectral Distance are defined below.

An audio waveform is a (1,N) matrix consisting of the amplitude of pressure waves at N discrete time steps. The number of time steps in one second is known as the sampling rate and is commonly 16 kHz. Audio is commonly visualized with spectrograms, which are 2D plots with a temporal and spectral axis. The spectral axis contains the intensity of frequency buckets at different time slices on the temporal axis. In other words, the spectrogram depicts the frequency decomposition of a signal as a function of time. The conversion from waveform to spectrogram is handled with a short-time Fourier transform (STFT). Specifically, the spectrogram is simply the log of the absolute value of the STFT.

Compressed or upscaled audio is often evaluated with respect to a few error metrics. The ones used in this paper are Peak Signal-to-Noise ratio (PSNR) and Log-Spectral Distance (LSD).

Given a ground-truth signal x and an altered signal y, the peak signal to noise ratio is calculated as follows, where the operand of the log is the mean-squared error of each sample in the waveform with respect to the groundtruth.

$$\text{PSNR}(x, y) = -10 \log\|x - y\|_2^2 \tag{1}$$

Normally this value would be subtracted from the 'peak' intensity of the ground-truth signal, however all signals in this work are normalized to have a peak value of 1, rendering that term to be 0. The log-spectral density is calculated in terms of the STFT's of the signals, $\hat{x}$ and $\hat{y}$, which consist of L time-windows and K frequency bins per FFT:

$$\text{LSD}(x, y) = \frac{1}{L} \sum_{l=1}^{L} \sqrt{\frac{1}{K} \sum_{k=1}^{K} \left(\log(|\hat{x}(l,k)|^2) - \log(|\hat{y}(l,k)|^2)\right)^2} \tag{2}$$

This boils down to the mean RMSE (root-mean-squared error) of the spectral power of the signal.

3 Prior Work in Audio Upscaling

The most basic form of upsampling uses a spline to interpolate between data points. However, this approach tends to miss the higher frequency sounds that people use to understand what is being said. This upsampling is often called bandwidth extension in literature.

Several different types of loss are used in audio super-resolution work. The most common loss is simply a L2 loss from high resolution training data and the generated super-sampling from low resolution data. This loss function has the advantage of being straight forward and easy to implement, however people have noticed that even well trained networks with this loss function do not sound "good" as measured by humans scoring soundtracks on a scale from 1 to 100 [1].

A popular approach is to use a fully convolutional network that starts with several layers of input down-sampling and convolving followed by several layers of upsampling and convolving. This architecture is called a U-net [2]. The U-net architecture struggles with the vanishing gradient problem. To solve this issue a residual network (ResNet) is used which combines the output of the upper layers with lower layers (the addition happens every other layer).This approach was used on the VCTK database to successfully super-sample sound clips [1]. The Kuleshov et al. paper [1] which used a ResNet is currently the state of the art and is highly cited on google scholar although this is being challenged by many groups including the lab that published the paper.

Another approach that has been used is generative adversarial network (GAN) based. GAN based approaches use a generator and a discriminator to generate super-sampled audio and a discriminator to determine the probability that a sample is original high-frequency data or generated high frequency reconstructions. This approach was used in 2019 by Kim [3]. However, the results were not noticeably improved from the results by Kuleshov.

Finally, recurrent neural networks (RNN) have been used as well since time series data such as audio signals are often modeled with RNNs. A recent paper in 2020, used a combination of a RNN and a ResNet (specifically in each block within a ResNet a RNN was added) to complete audio super-resolution tasks. It had similar performance to Kuleshov et al [4].

4 Proposed method

Our proposed method to developing a robust audio-upscaling solution is two fold. We first determine what features make low quality audio less desirable by recording sound clips on an assortment of microphones of varying quality and compare their responses. Using this information, we can correctly augment our high quality training data to match the response from a low quality microphone. This will help us determine if our approach is robust against the additional noise, reverberation, and compression associated with low quality audio. Next, we construct a neural network to convert this low quality audio to higher quality audio. This is accomplished with a novel frequency-fused ResNet (FFRN) and generative adversarial network (GAN). The next few subsections outline how we tackled these two concepts.

4.1 Data Collection

The previous work in audio ”super-resolution” [1] sought to up-scale audio with a low resolution (low sampling rate) into a higher resolution (higher sampling rate), however this project’s goal is to improve the quality overall of the input signal which in practice can have ”quality” issues other than just a low sampling rate. One of the most common problems, for instance, is background noise, which can come as the result of a variety of short-comings in a microphone setup.

Because there are so many different avenues which can bring rise to ”poor” microphone quality, a substantial amount of example data needs to be recorded, pairing the recording of the same voice on low and high quality microphones simultaneously (see Figure 5). To train a network for ANY low quality microphone would be a massive undertaking in terms of data collection and is out of scope for this project.

The goal instead is to leverage the pre-existing Voice Cloning ToolKit (VCTK) dataset which is comprised of high fidelity recordings of more than 100 different English speakers reading sentences taken from news articles. Two different methods are considered in generating low-quality versions of those provided by the VCTK dataset.

The first method is to play the recordings out of a high-quality electronic speaker and re-record the data on one or more low-quality microphones. The advantage of this method is that real microphones are being used to inject the low-quality microphone properties into the data. The disadvantages are that the speaker properties are not easily accounted for, that this method needs to be done in real-time and serially, and that the VCTK dataset consists of over 44 hours of voice recordings. The second method is to manually analyze the difference between a low-quality versus high-quality simultaneous recording of a voice and determine a set of automated transformations which add the low-quality properties to the VCTK dataset directly. Some of these transformations might be adding random artificial environmental noise, or clipping the peaks of the sound-waves to a maximum amplitude. This allows for generation of much more data, at the cost of it being ”artificially” low-quality, and depends on a human’s characterisation of the low-quality microphone recordings.

The second method was chosen as it was more feasible and allowed for trying many different transforms. Three different transforms were attempted:

1. White noise: A 10 second clip of silence was recorded on a low-quality microphone as a white-noise sample. Random segments of this clip were automatically added to the sound clips in the VCTK dataset at a specified signal-to-noise ratio.
2. Room Reverberation: A recording of a hand-clap from the low-quality microphone was edited to align the start of the clap at precisely the start. A 1-D convolution is then performed with this signal as the kernel on the dataset to simulate the reverb effect.
3. Compression: A lossy compression codec is applied to the sound and then reverted back to the raw format to simulate the effect of a lower bitrate signal.

4.2 Waveform ResNet

Our waveform model is inspired by the network developed by Kuleshov et al.[1] which can be broken down into three steps: downsampling, bottleneck, and upsampling as shown in Figure 1.

Downsampling is achieved by repeated convolutions at a stride of two. This stride removes half the data for each subsequent downsampling layer which greatly reduces the computation time compared to fixed size layers and provides the ability to convolve larger lengths of audio. That is, the earlier downsampling layers can learn short-term features while later layers deeper in the network learn long-term features of the audio data. Additionally, a rectified linear unit (ReLU) was used after each convolution to induce non-linearity.

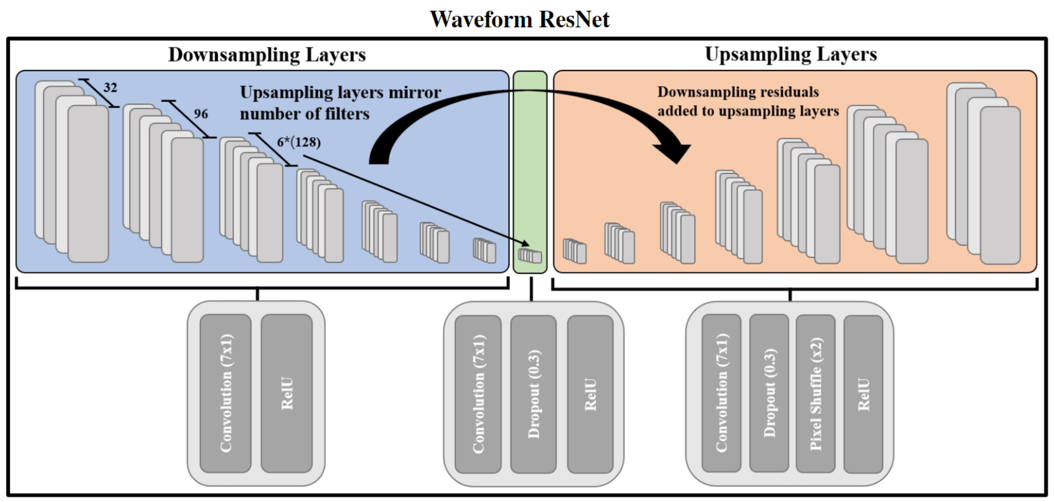

Figure 1: Waveform residual neural network model. Note that the last upsampling layer does not contain ReLU as that wouldn't allow for negative outputs

The bottleneck sequence, as shown in Figure 1, is the transition from the downsampling layers to the upsampling layers. It acts similarly to the downsampling layers except for the introduction of dropout regularization. This dropout layer is also used in the forthcoming upsampling layers and helps keep the model from overfitting to specific sound clips in our dataset.

The upsampling sequence acts similarly to the inverse of the downsampling sequence except for one key difference: pixel shuffling to increase output size instead of convolution strides of 2 to decrease output size. Pixel shuffling is a method of reshaping an output to half the number of filters and double the input data size. This was originally used in the space of image super resolution [5] and adapted to work on one dimensional audio data for our needs. Additionally, there are residual connections between downsampling and upsampling layers which provides more information in reconstruction that increases model stability and training speed. Lastly, ReLU was used after each convolution to induce non-linearity except the last layer since the output should not be bounded by the activation layer.

Hyperparameters were tuned based on performance results in training and validation loss. Specifically, all convolutional layers use a filter length of seven. The filter length does not need to change because the filters act on varying temporal scales as the data travels through the network. The number of filters per layer is displayed in Figure 1 and was found to be the minimum number of layers necessary before the loss increases. The number of upsampling and downsampling layers is currently set to seven as testing suggested that more layers didn't improve performance. Finally, a dropout probability of 0.3 was used to strike a compromise between overfitting and increased learning time. These values, along with the standard parameters used in our Adam optimizer, are summarized in Table 4 in the Appendix.

4.3 Frequency mini-ResNet

The frequency mini-ResNet operates very similarly to our waveform ResNet, however there are several key differences between the two. The largest difference is that the frequency ResNet operates on 2D data from a STFT compared to the 1D data in waveform ResNet. This 2D data is the frequency and phase information of our input, modeled as complex numbers, at different slices of time. This leads to another difference: the layers in frequncy ResNet operate in the domain of complex numbers. This is achieved by splitting the real and imaginary portions of each element into two separate matrices. Another difference is that the frequency ResNet has only three upsampling/downsampling layers compared to the seven in the waveform ResNet. This is because the layers operating on 2D data scale the size by a factor of four compared to the factor of two when operating on 1D data. On a final note, the filters for the convolutional layers in the frequency ResNet are uniquely shaped as (3,11) where we operate across 3 units of time and 11 units of frequency. This filter is longer

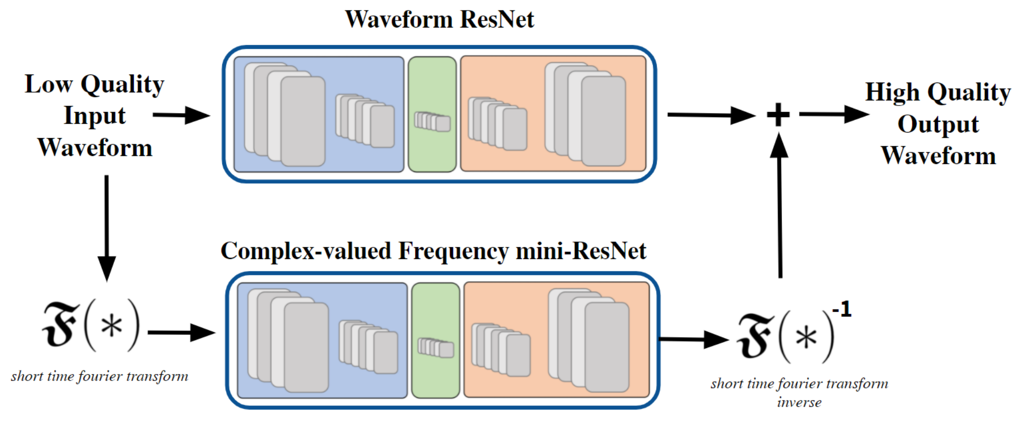
Waveform ResNet
Low Quality Input Waveform
High Quality Output Waveform
Complex-valued Frequency mini-ResNet
$\mathfrak{F}(*)$
short time fourier transform
$\mathfrak{F}(*)^{-1}$
short time fourier transform inverse

Figure 2: Combined frequency and waveform residual neural network model (FFRN). The complex valued frequency mini-ResNet is similar to the waveform ResNet, however it has fewer downsampling and upsampling layer and the layers are modified to handle complex numbers

in frequency compared to time because the patterns between low and high frequency span a larger length compared to the temporal patterns.

4.4 Frequency Fusion ResNet

The waveform ResNet and the frequency mini-ResNet are fused together to form the novel frequency fusion ResNet (FFRN). The waveform ResNet provides decent results in upscaling audio, but lacks in the high frequency details. The frequency min-ResNet, on the other hand, can easily identify patterns between low and high frequencies since it operates in this domain. The intuition behind FFRN is to use the strengths of both models where the waveform ResNet acts as the course adjustment and the frequency ResNet acts as the fine adjustment. These models are fused by converting the input waveform into the frequnecy domain with a STFT which is then used by the frequency ResNet, and the result is converted back into a waveform with an inverse STFT. This is shown in Figure 2 and in Equation 3 below:

$$F(x) = W(x) + \mathfrak{F}^{-1}(S(\mathfrak{F}(x))) \tag{3}$$

Where x is the input waveform, $W()$ is the waveform network, $\mathfrak{F}()$ is the STFT, $S()$ is the frequency network, and $F()$ is the output of FFRN. Other methods to view the frequency decomposition of an audio signal such as the spectrogram, cepstrum, or MFCC were not used because these operations are non-invertible; that is, we cannot reconstruct the waveform after applying these transformations.

4.5 Generative Adversarial Network

State-of-the-art solutions in image super resolution use generative adversarial networks (GANs) to create better, more human-centered loss criterion and we implemented similar techniques for audio upscaling [6]. The GAN used the waveform ResNet as the generator function. The discriminator function used is based on a current implementation of a DCGAN for image generation. It takes audio as a 1xN input and outputs a scalar probability that the audio generated is a high quality transform. The audio is processed through a series of 1-D convolutions, batch normalization, and leaky ReLU, and outputs the final probability through a Sigmoid activation function. The archicture can be updated with more layers if necessary. The DCGAN paper [7] mentions it is a good practice to use strided convolution rather than pooling to downsample because it lets the network learn its own pooling function. Also batch normalization and leaky ReLU functions promote healthy gradient flow which is critical for the learning process of both generator and discriminator based on the original GAN paper by Goodfellow et al, [7]. Let $D(x)$ be the discriminator network which outputs the (scalar) probability that x came from training data rather than the generator. $D(x)$ should be high if x comes from training data and low if it comes from the generator. Let z be a latent space vector sampled from a standard normal distribution. $G(z)$ represents the generator function which

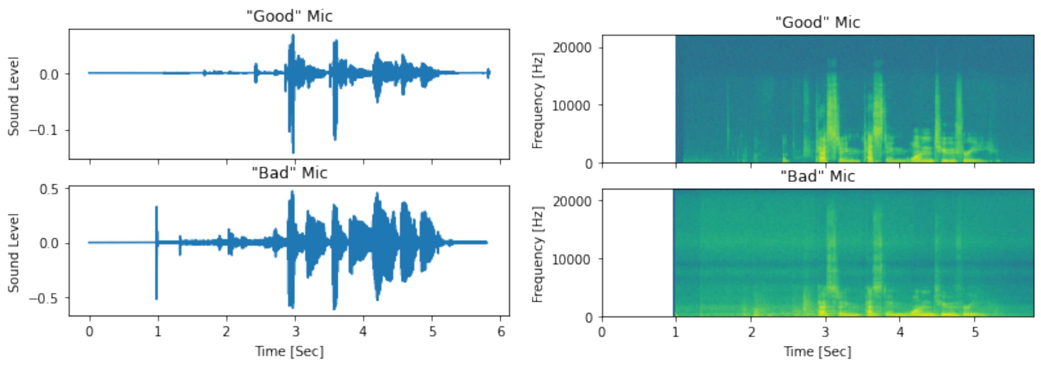

Figure 3: Waveforms and frequency responses of recording the same voice (”tes-ting, tes-ting, one, two, three) on two different microphones simultaneously. The ”bad” microphone was the internal microphone of a laptop while the ”good” microphone was a headset mic.

Table 1: PSNR and LSD Dataset Comparison (Higher PSNR and Lower LSD is better)

Dataset Modification	White Noise	Room Reverb	Lossy Compression
Input LSD	3.019	4.027	0.077
Upscaled LSD	3.357	3.319	1.086
Input PSNR	24.627	20.241	78.485
Upscaled PSNR	33.379	31.352	46.995

maps the latent vector z to data-space. $G(X)$ estimates the distribution that the training data comes from to generate fake samples from that estimated distribution (p_g). The GAN loss function used is shown in Equation 4 by Goodfellow et al, [7].

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z))] \tag{4}$$

5 Experimental Results

The following section displays our experimental results. Alternative data collection and low quality audio generation methods were explored and their effects are shown. Several implementations were added to improve the existing results. The FFRN and GAN were implemented and evaluated using the criteria stated in Section 2. A spectrogram was generated for one sample for each of the networks. Ten samples of high quality sound can be found **here**. The low quality (down-scaled) version of those samples can be found **here**. The Kuleshov et al. implementation results upscaled from the low quality audio can be found **here**.

5.1 Data Collection and Low Quality Audio Methods

Figure 3 shows the results of one of these sample recordings on two different microphones. From observation alone, there are two key properties that can be gleaned which explain the ”bad” microphone's worse quality. There is more white noise in the ”bad” mic, which can be observed best in the uniform streaks in the frequency response. By analyzing the frequency response of the quiet portions of that signal, it should be possible to replicate this white noise artificially.

The other key difference is that the ”bad” mic has more echo, which can be observed by listening to the sample itself and also in the waveform (note the longer time it takes for the signal to dampen after the peak in each syllable). To simulate this, the Room Impulse Response was sampled and applied to the high-fidelity sound, as a convolution in time [8].

Datasets using various combinations of added white noise, room reverb, and lossy compression were made and trained on the waveform ResNet model. See Table 1 for a summary of the results. When upscaling the samples with white noise, the model failed to eliminate the noise and simply modified the volume of the signal. When applied to room-reverb, though resulting sound was very

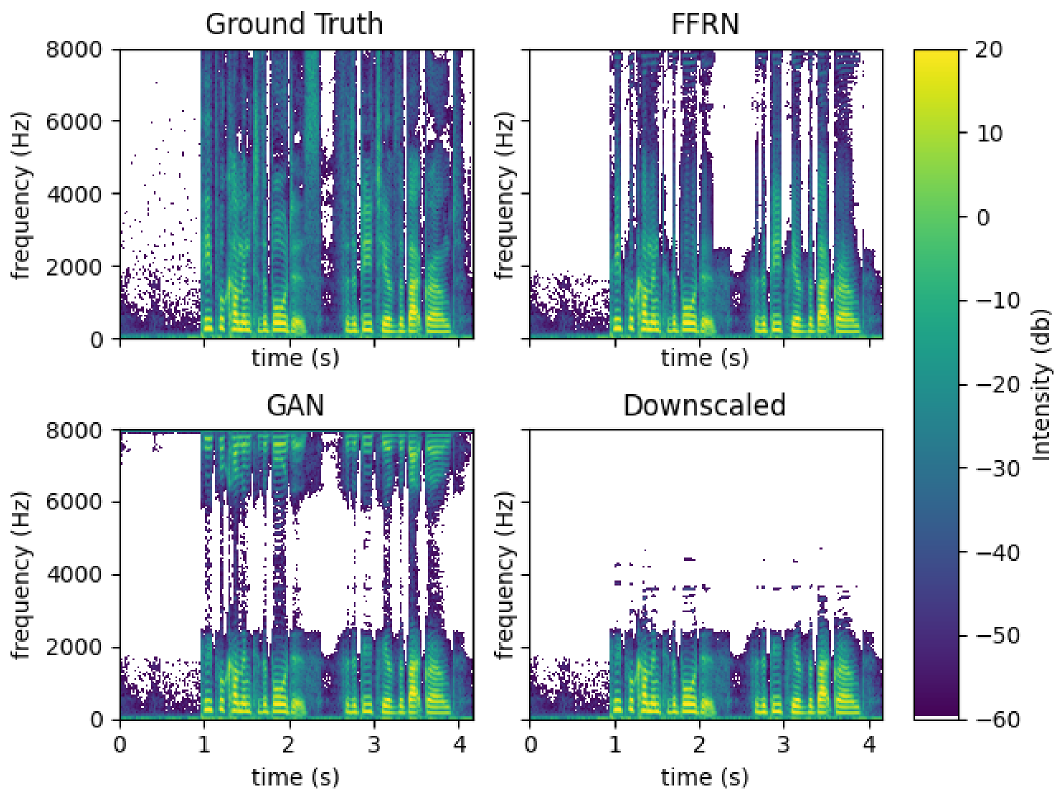

Figure 4: Spectrogram of the original high resolution audio (top-left), low quality input (bottom-right), GAN output (bototm-left), and FFRN output (top-right).

degraded and did not sound like the original, though the total loss was decreased. In the case of the compressed audio, the upscaled audio had worse objective metrics than the original compressed audio.

5.2 GAN Results

While there has been significant work showing that GAN have been able to sharpen features in image resolution, the audio output from the network shows significantly worse performance than the FFRN Implementation. Figure 4 shows the spectrogram of the original audio, GAN output, and low quality input. The GAN ouput does show amplified high frequencies on the spectrogram. Despite having the loss for the discriminator and generator converge on each other during training, the high frequencies are not accurate and cause the audio to sound worse than the results from the FFRN implemention and the original Kuleshov et al. implemention. The audio can be found **here**. The low spectral density and signal to noise ratio results can be found in Table 2.

5.3 Frequency Fusion ResNet Results

The results from the frequency fusion ResNet showed significant improvement over the GAN network and score higher than the original Kuleshov et al. implementation. These results can be found in Table 2 using the criteria described in Section 2. Additionally, the spectrogram shown in Figure 4 displays the results compared to the original high quality audio, low quality input, and the GAN upscaling method.

Several comparative audio tests were run with human test subjects to quantitatively assess the relative audio quality. The Kuleshov et al. paper[1] uses a MUSHRA (MUltiple Stimuli with Hidden Reference and Anchor) test to assess the audio quality. MUSHRA tests require trained listeners to rank audio on a scale of 1 to 100 as specified in a 36 page international telecommunications union

Table 2: PSNR and LSD Algorithm comparison averaged over 11 different audio samples (higher PSNR and Lower LSD is better)

Algorithm	Kuleshov	FFRN	GAN	Spline
LSD	3.436	**2.859**	7.719	11.763
PSNR	44.706	**46.317**	19.391	40.066

setup document. There were not resources to conduct this exact test, so an alternative was devised. A high resolution target sound clip was played for a test subject followed in random order the Kuleshov result and the FFRN result. Test subjects were asked to which clip better matched the target clip. Participants ranked the FFRN clips as better slightly more often, although not in a statistically significant way. Participants were also played the low resolution clip and the corresponding FFRN upsampled clips (again with random ordering) and asked which sounded better. The test subjects overwhelmingly preferred the FFRN clips. These results are summarized in Table 3. The upscaled results from the trained Frequency ResNet model can be found **here**.

Table 3: Results from human subjective testing comparing our results to the current state-of-the-art and classical upscaling methods across 11 different audio clips.

	Participants				
	1	2	3	4	Preference (%)
Cubic Spline	0	0	0	0	0
Kuleshov et al.	5	8	3	3	**47.5**
FFRN (Ours)	5	2	7	7	**52.5**

5.4 Frequency Fusion ResNet Complexity

The goal of this project is to create a solution for video conferencing. Therefore it is imperative our model is capable of running in real-time on consumer hardware. We have achieved this with a 1.4±0.1s runtime on a 12s audio clip running on a NVIDIA GTX 1660s. More impressively, we achieved real-time results on the CPU with a 4.2±0.2s runtime on the same 12s audio clip.

Furthermore, our FFRN model is more compact than the current state-of-the-art solution as ours has 21.7 million learnable parameters while the current state-of-the-art solution has 62.2 million learnable parameters.

6 Conclusion

We have demonstrated progress towards audio super resolution using a FFRN on the VCTK dataset. Our real-time capable implementation scores better than the current state-of-the-art solution in objective metrics, but does not show a statistically significant improvement with subjective metrics. Additionally, we have also shown that these models don't perform well against more realistic examples of poor mic quality audio, and future work should be done in models which can account for the distortions due to room reverberation. Finally, further work will also focus on human perceptual loss function incorporation and ideally show better performance.

7 Author Contributions

Vishnu Rengaraj and Adam Ingerman worked on implementing the paper results in PyTorch, creating the spectrograms, audio samples, and calculated the current loss. Vishnu Rengaraj worked on the fusion frequency ResNet implementation. Paul Knudsen worked on replicating the Kuleshov results with their code, wrote the initial version of the GAN model, and completed the prior work review. Adam Ingerman worked on the final GAN implementation. Eric Ward worked on dataset manipulation, debugging, running models, and researching alternative methods. All co-authors were involved in writing this report. All co-authors equally contributed to this project.

References

[1] V. Kuleshov, S. Z. Enam, and S. Ermon, “Audio super resolution using neural networks,” 2017.

[2] S. Sulun and M. E. P. Davies, “On filter generalization for music bandwidth extension using deep neural networks,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 1, p. 132–142, Jan 2021. [Online]. Available: http://dx.doi.org/10.1109/JSTSP.2020.3037485

[3] S. Kim and V. Sathe, “Adversarial audio super-resolution with unsupervised feature losses,” 2019. [Online]. Available: https://openreview.net/forum?id=H1eH4n09KX

[4] S. Birnbaum, V. Kuleshov, Z. Enam, P. W. Koh, and S. Ermon, “Temporal film: Capturing long-range sequence dependencies with feature-wise modulations,” 2020.

[5] C. Dong, C. C. Loy, K. He, and X. Tang, “Image super-resolution using deep convolutional networks,” 2015.

[6] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, “Photo-realistic single image super-resolution using a generative adversarial network,” 2017.

[7] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” 2014.

[8] B. Waslo. (1993, June) Maximum length sequence (mls) based measurements with laud. [Online]. Available: http://www.libinst.com/mlsmeas.htm

A Supplementary Tables and Figures

Table 4: List of hyperparameters tuned during testing.

Hyperparameter	**Value**
Filter Length	7
Downsampling/Upsampling Sequences	7
Dropout Probablity	0.3
Learning Rate (Adam)	0.001
Betas (Adam)	0.9,0.999

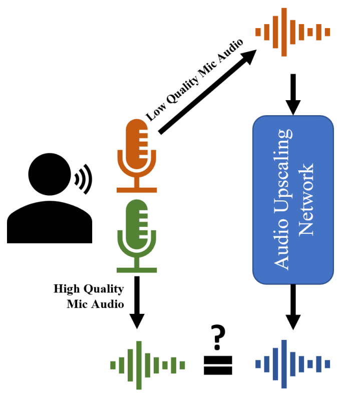

Figure 5: Hypothetical training pipeline