– Independent Work Report Spring, 2018 –

Hierarchical Recurrent Neural Networks for Audio Super-Resolution

Berthy Feng
Advisor: Adam Finkelstein

Abstract

This work proposes a recurrent model for audio super-resolution, the task of inferring the high-resolution version of a low-resolution recording. Given the lack of baseline methods and ambiguity regarding the most appropriate deep learning method for this task, we focus on recurrent neural networks. We propose a hierarchical recurrent neural network (HRNN), trained using a loss function that combines a regression-based loss and a perceptual loss. In this paper, we present the baseline HRNN architecture, as well as two types of perceptual loss to improve on the baseline. Our baseline model outperforms interpolation methods, and our proposed perceptual losses improve on the baseline by providing explainability and producing perceptually realistic results.

1. Introduction

1.1. Audio Super-Resolution

The task of audio super-resolution is to improve the resolution of an audio recording. There are various ways to conceptualize this task, whether in the time domain or in the frequency domain.

In the time domain, the task is to increase the sampling rate of a given audio waveform. For example, audio sampled at 16 kHz has twice the resolution of audio sampled at 8 kHz. The higher sampling rate captures more samples per time unit, thus capturing higher-frequency sound components, which might include the overtones of voiced speech and the reverberations of the room. In the frequency domain, the task is to expand the bandwidth of an audio signal. Also known as bandwidth extension, the goal is to produce a wideband signal from a given narrowband signal, where the wideband output has the energy at the higher band frequencies restored.

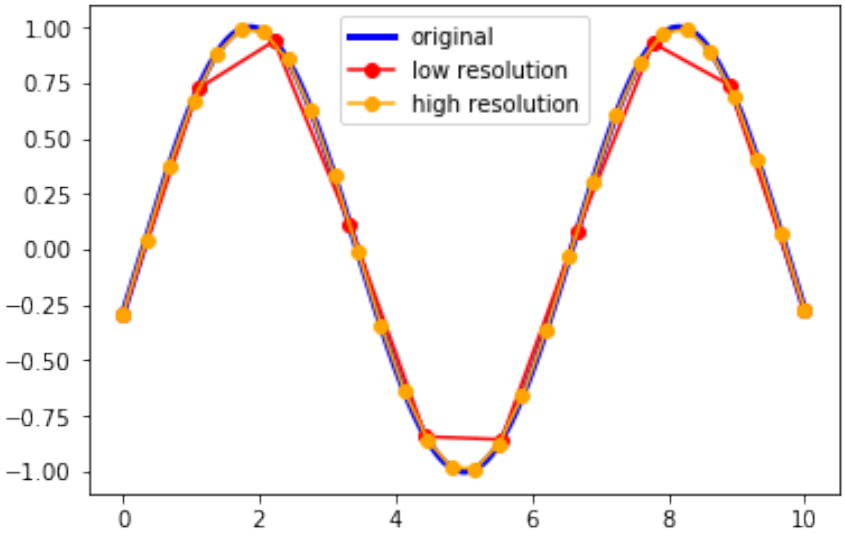


Figure 1: Visualization of the effect of sampling rate on waveform resolution. The orange line samples the original sine wave three times as frequently as the red line, resulting in a more accurate waveform. Our task is to infer the orange line given only the red line.

Both ways of thinking about audio super-resolution are valid, as they both aim to restore high-frequency components. However, they often lead to different results. A time-domain approach models audio as a time series, thereby more accurately reproducing a signal's waveform and phase information. A frequency-domain approach models audio as a spectrum of frequencies, thereby more accurately reproducing the energy at each frequency and the "color" of the audio. In other words, a time-domain approach might lead to a more realistic waveform, whereas a frequency-domain approach might lead to a more realistic spectrogram. Currently, most state-of-the-art audio synthesis methods operate in the frequency domain, utilizing a feed-forward, rather than recurrent, network.

1.2. Our Work

Our approach is in the time domain, as the proposed architecture operates at the waveform level. Aiming for accuracy in the frequency domain, as well, we propose a perceptual loss to encourage realistic spectrogram results. Our task specifically is the super-resolution of single-speaker speech, where the input is an 8 kHz speech recording, and the output is the 16 kHz version with the higher band frequencies restored.

The reason for single-speaker is that it is difficult to build a model representing all human voices, since a voice is characterized by its timbre. The reason for increasing the sampling rate to 16 kHz is to lend a sense of presence to the recording. Many bandwidth extension techniques aim to go from a very low sampling rate, such as 3 kHz, to a medium sampling rate, such as 8 kHz. Going to 8 kHz restores basic speech quality and is useful in many telecommunications applications to make transmitted speech distinguishable. However, the frequencies above 8 kHz add more subtle resolution and color, giving the listener a clearer picture of the room and speaker.

Given the sequential nature of audio waveforms, we experiment with a recurrent model for audio super-resolution. We focus on hierarchical recurrent neural networks, which are able to model sequential information at multiple timescales.

1.3. Recurrent Neural Networks

A recurrent neural network (RNN) is a neural network with feedback connections [15]. At a given time step t, the RNN has a hidden state, h_t, which is calculated as a function of the previous hidden state h_{t-1} and the input specific to that time step, x_t. This recurrence formula is expressed as:

$$h_t = f_W(h_{t-1}, x_t) \tag{1}$$

The function f_W with parameters W, once learned, stays constant throughout all time steps.

RNNs model sequential data extremely well. They are widely used in tasks related to natural language processing, including machine translation, image captioning, and text generation. Since their popularity soared in the 1990s [15], various types of RNNs have emerged, including long short-term memory (LSTM) networks [4] and gated recurrent units (GRU) [1].

1.4. Hierarchical RNNs

The sequential nature of audio lends itself to a recurrent model. However, one RNN might not be sufficient to capture the necessary information to reconstruct a high-resolution signal. Audio contains low-level information, such as the exact sample values, as well as higher-level information,

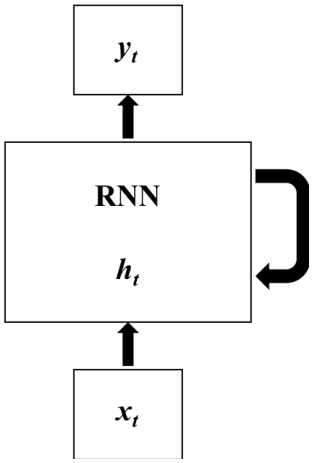


Figure 2: RNN visualization.

such as speech phonemes, echoes, and reverberations. Hierarchical RNNs (HRNNs) offer a way to use information at multiple levels.

An HRNN contains multiple tiers of RNNs, where each tier's input covers a different scope. SampleRNN is a state-of-the-art method for audio generation whose architecture is an HRNN [12]. A SampleRNN with three tiers, for example, might have the top-tier RNN process eight samples at a time, the middle-tier RNN process two samples at a time, and the low-tier RNN process one sample at a time. In an HRNN, the lower-tier RNNs typically take the higher-tier RNN outputs as additional inputs. Arranging RNNs in this hierarchy allows a model to learn from multiple levels of information. This approach would be especially useful in audio, where there is a hierarchy of recorded speech features.

1.5. Outline

In Section 2 we describe the motivations of our work, including the applications of audio super-resolution and the inspirations behind our ideas for using a recurrent model and a perceptual loss. In Section 3, we state our two primary and interrelated goals of the project. In Section 4, we review related work in bandwidth extension and audio generation. In Section 5, we explain the proposed HRNN architecture and the two types of perceptual loss. In Section 6, we describe our implementation and training methods. In Section 7, we present our results and demonstrate that our proposed method achieves perceptually realistic output. In Sections 8 and 9, we offer conclusions

and directions for future work.

2. Motivation

2.1. Audio Super-Resolution Applications

Traditionally, audio super-resolution has been known as bandwidth extension. The task had its primary applications in telecommunications, where it was important to improve the quality of speech recorded on low-bandwidth devices. Audio super-resolution has more modern applications, as well. It is useful in text-to-speech synthesis, where speech might be synthesized first at a low resolution. It is also useful for people who wish to improve the quality of their audio recordings, either because the audio was originally recorded on a low-quality microphone or because the audio was compressed to a low sampling rate. Since our work focuses on the high bands above 8 kHz, our goal is to give a sense of presence to a speech recording. We assume that the narrowband speech is already intelligible, and our aim is to improve the listening experience.

2.2. Understand RNN Modeling Capabilities

The success of WaveNet [17], a feed-forward network for audio generation, has prompted many researchers to default to a feed-forward network for audio synthesis tasks. However, the possibilities of a time-based, recurrent approach to audio super-resolution have yet to be fully explored. The need for experimentation with RNNs for super-resolution is a motivation for this work.

2.3. Inspirations from Computer Vision

Many ideas in computer vision are transferable to audio. There have been multiple state-of-the-art methods proposed for image super-resolution, so an insight from image super-resolution could help us in the audio domain. Specifically, the addition of a perceptual loss helps significantly in image super-resolution. The key idea of a perceptual loss is to encourage perceptually realistic results, and the aim of audio super-resolution is also to produce perceptually realistic results (i.e., transformations such as phase shift do not impact the quality as perceived by the human ear, so we

are not concerned with exactly matching the ground truth). Given this promising idea of a perceptual loss, it might be possible to overcome the typical drawbacks of an RNN, such as oversmoothing. Thus another motivation is to utilize the insights made in computer vision and apply them to an audio problem.

3. Goals

The goal of this project is to develop a hierarchical recurrent neural network for audio super-resolution. Specifically, this means accomplishing these two steps:

1. Propose a baseline HRNN method that achieves acceptable results on single-speaker speech.
2. Propose a perceptual loss to improve the baseline method.

The more general goal is to identify the advantages and disadvantages of an RNN-based approach to audio super-resolution and suggest a direction for future work.

4. Related Work

4.1. Learning-Based Vocoder Approaches

Vocoder methods are a popular way of modeling audio in the frequency domain. In this approach, the spectral parameters of the narrowband input signal are extracted and mapped to the spectral features of a corresponding wideband output signal. Many classical machine learning-based methods exist, including codebook mapping, Gaussian mixture models, and hidden Markov models. Neural networks have also been developed for encoding input features and decoding them to produce wideband output. Both dense neural networks [10] and recurrent neural networks [3] have been proposed for this. The difficulty with these approaches is restoring the phase spectra, as well as maintaining the original speech quality despite the parameterization of the vocoder.

4.2. Generative Audio Models

Waveform-based audio generation methods have become increasingly popular as a result of the success of WaveNet [17] and its successor, Tacotron 2 [14]. Both use dilated convolutions to

expand the receptive field of the convolutional neural network (CNN) and are fully feed-forward. Tacotron 2 is conditioned on the mel-spectrogram predictions, an idea that influences our design of a spectrogram-based perceptual loss. WaveNet and Tacotron 2 are used for random audio generation and text-to-speech synthesis, both problems that require modeling the distribution of audio from scratch. However, these methods are not optimized for the super-resolution task, which allows for more supervision during training. While these feed-forward networks achieve state-of-the-art synthesis, our aim is to propose an alternative, recurrent network that specializes in super-resolution.

SampleRNN [12] is the leading recurrent generation model and is the architectural inspiration for our proposal. Again, however, it is not optimized for super-resolution, as the intention of SampleRNN is to model the distribution of audio from scratch and allow for randomness in the output. Super-resolution, however, is conditioned on the input signal, so the task allows for a more restricted output space and a supervised training scheme.

4.3. Deep Learning Approaches

Only recently have researchers designed deep learning models specifically for audio super-resolution. Kuleshov et al. propose a simple convolutional neural network as a baseline neural net for audio super-resolution [8]. Ling et al. apply SampleRNN to bandwidth extension and claim better results than a fully feed-forward network [11]. Both works present fairly simple models and suggest opportunities for improvements. The work of Ling et al. provides the inspiration for our method, although we modify and improve their model to make it more suitable for audio super-resolution.

5. Approach

5.1. HRNN Architecture

In general, a hierarchical recurrent neural network may have K tiers in total, where the top $K-1$ tiers operate on a frame level (where a frame is a sequence of samples), and the bottom tier processes one sample at a time. We propose as our baseline model a three-tier hierarchical recurrent neural network, which includes two frame-level tiers and one sample-level tier.

Input. The narrowband input is sampled at 8 kHz and then upsampled to 16 kHz so that it shares the same sample length as the wideband, 16 kHz output. Note that even though the input contains 16000 samples per second, its frequency content still does not exceed 8 kHz. Each sample is then quantized using mu-law encoding to a value between 0 and 255. This allows us to make each input sample a one-hot encoding vector of length 256, where the index corresponding to the sample's quantized value is marked "1" and all other indices are marked "0". Let $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$ denote the entire input sequence of length T (e.g., for three-second audio sampled at 16 kHz, $T = 48000$). Each $\boldsymbol{x}_t$ denotes the one-hot vector representing the quantized value of the sample at t.

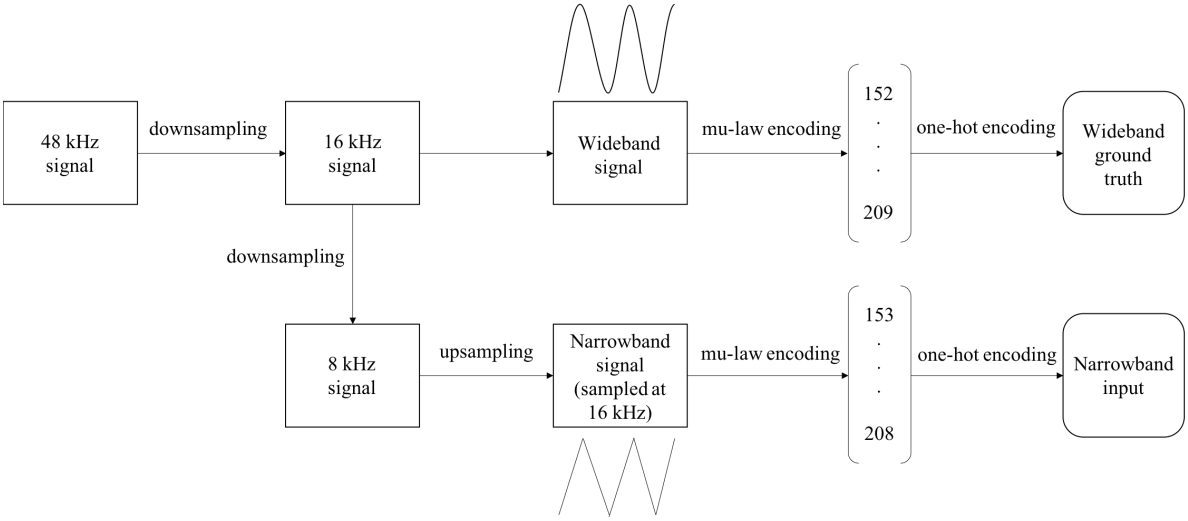


Figure 3: Input preparation.

Tier 3. The highest frame-level tier processes eight samples at a time. We implement this as a single-layer RNN (although any number of layers can be added). At time t, its input is:

$$\boldsymbol{f}_t^3 = \{\boldsymbol{x}_t, \ldots, \boldsymbol{x}_{t+7}\} \tag{2}$$

The hidden state of the RNN is formulated as:

$$\boldsymbol{h}_t^3 = g^3(\boldsymbol{h}_{t-1}^3, \boldsymbol{f}_t^3) \tag{3}$$

where $\boldsymbol{h}_{t-1}$ is the previous hidden state, and g^3 is the learned RNN function.

To pass information onto the next tier, the top tier generates conditioning vectors for Tier 2. Since Tier 2 processes two samples a time and Tier 3 processes eight samples at a time, Tier 3 outputs four conditioning vectors at each time step (one for every two samples it processes). The conditioning vectors at time t are formulated as:

$$\boldsymbol{d}_{t+j}^3 = \boldsymbol{W}_j^3 \boldsymbol{h}_t^3, j = 0,2,4,6 \tag{4}$$

All $\boldsymbol{d}_{t+j}^3$ become the input for Tier 2.

In Tier 3, we increment t by eight samples for each new time step. This is because we want each time step for Tier 3 to have non-overlapping input sequences.

Tier 2. The middle tier processes two samples at a time. Also implemented as a single-layer RNN, its input at time t is a linear combination of the sample inputs and a conditioning vector from Tier 3:

$$\boldsymbol{f}_t^2 = \{\boldsymbol{x}_t, \boldsymbol{x}_{t+1}\} + \{\boldsymbol{d}_t^3\} \tag{5}$$

Its hidden state is expressed as:

$$\boldsymbol{h}_t^2 = g^2(\boldsymbol{h}_{t-1}^2, \boldsymbol{f}_t^2) \tag{6}$$

Tier 2 also outputs a conditioning vector for each sample, which is in turn used as input for Tier 1. At every time step t, the RNN outputs two conditioning vectors:

$$\boldsymbol{d}_{t+j}^3 = \boldsymbol{W}_j^3 \boldsymbol{h}_t^3, j = 0,1 \tag{7}$$

In Tier 2, we increment t by two samples for each new time step so that the input sequences are non-overlapping.

Tier 1. The lowest tier, also known as the sample-level tier, is a fully feed-forward multilayer perceptron (MLP). We implement it as three fully connected layers. At this stage, the input samples are sent to a learned 256-dimensional embedding layer, so each input sample $\boldsymbol{x}_t$ is embedded as the real-valued, 256-dimensional vector $\boldsymbol{e}_t$. The input, denoted as $\boldsymbol{i}^1$, is a linear combination of the

embedding vectors $\boldsymbol{f}^1 = [\boldsymbol{e}_1, \ldots, \boldsymbol{e}_T]$ and the conditioning vectors $\boldsymbol{d}^2 = [\boldsymbol{d}_1^2, \ldots, \boldsymbol{d}_T^2]$:

$$\boldsymbol{i}^1 = \boldsymbol{W}^1 \boldsymbol{f}^1 + \boldsymbol{d}^2 \tag{8}$$

For each input sample $\boldsymbol{x}_t$, the MLP predicts the corresponding output sample y_t in the narrowband version of the input signal (note that y_t is a scalar). In summary, we have described the architecture behind the function modeled by the HRNN:

$$\boldsymbol{y}_t = \text{HRNN}(\boldsymbol{x}_t), \forall t \in T \tag{9}$$

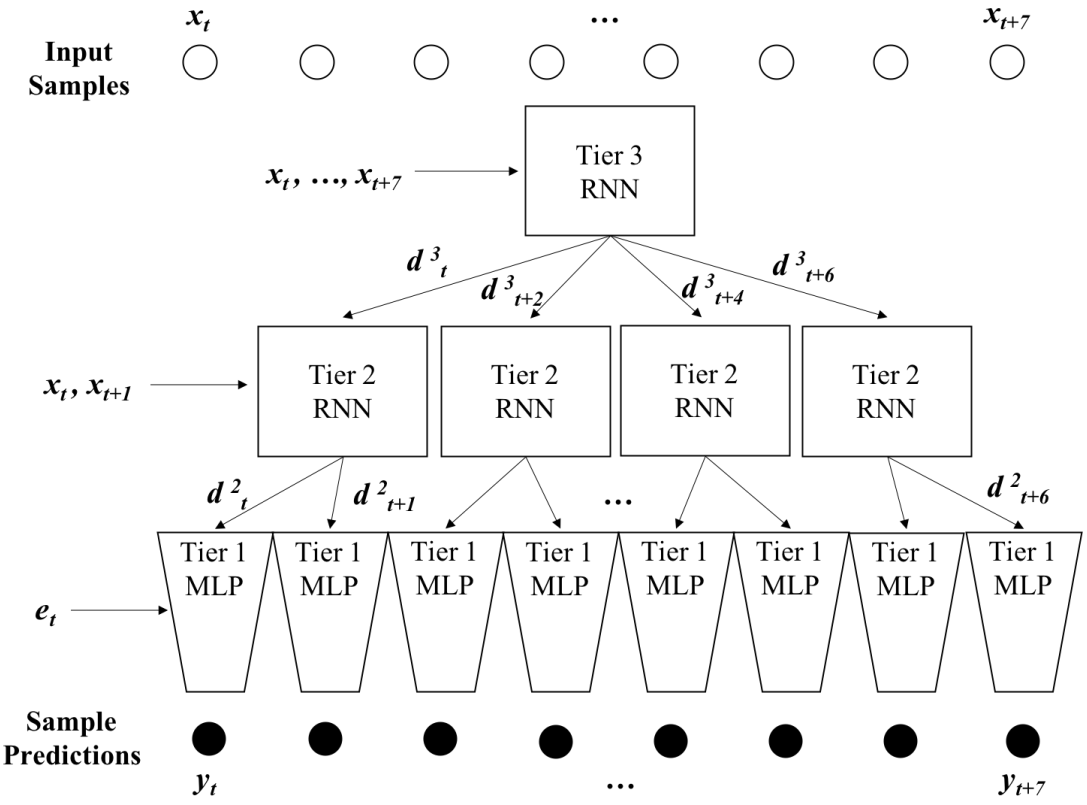


Figure 4: HRNN architecture.

5.2. Regression vs. Classification

We modify the SampleRNN architecture [12] and HRNN architecture of Ling et al. [11] by converting the function of the model from classification to regression. The previous HRNN models output a probability distribution over all quantization values at each time step. That is, for each input sample x_t, the HRNN outputs the conditional distribution $p(y_t|x_1, x_2, \ldots, x_t)$ (the distribution is actually also conditioned on samples after t, given the wider frame of Tier 1). Classification makes more sense for random audio generation than for super-resolution. SampleRNN was originally designed for unconditional audio generation, where the goal is to produce audio that sounds as if it were randomly drawn from a natural distribution. However, super-resolution is conditioned on the narrowband input signal and does not need randomness in the resulting samples.

Furthermore, converting the model to a regression-based one makes the addition of a perceptual loss much easier. If the output is a probability distribution, then to construct the output waveform, we must select a discrete value at each time step. This sample selection step is not differentiable, since the argmax or random choice from a distribution is discrete. Therefore, if we were to add any network onto the HRNN, the model would not be fully differentiable. This leads to issues in training, since the HRNN learns its parameters by backpropagation, which depends on the differentiability of all operations in the network.

By modifying the HRNN to perform regression instead, we make the model fully differentiable from the output waveform to the input waveform. We also reduce the model size, making it simpler and less memory intensive. This is an important way to optimize HRNN for audio super-resolution, since a regression-based model is more suited to the supervised nature of the task and enables more possibilities with additional losses during training.

5.3. Perceptual Losses

Inspired by the success of perceptual losses in image super-resolution, we experiment with the addition of a perceptual loss during the training of our HRNN. Johnson et al. prove the effectiveness of a perceptual loss in encouraging realistic results for image style transfer and super-resolution [6].

The research task for us is to design a perceptual loss that is suitable for audio.

The goal of the perceptual loss is to encourage perceptually realistic waveforms or spectrograms. Again, we are faced with working in either the time domain or the frequency domain. We design two perceptual losses: an adversarial loss for the time domain and a spectral loss for the frequency domain.

Adversarial loss. Generative adversarial networks (GANs) produce impressive results for image super-resolution [9]. We propose a GAN for audio super-resolution.

First proposed by Goodfellow et al., a GAN consists of a generator network and a discriminator network [5]. The generator attempts to produce realistic outputs, while the discriminator attempts to distinguish between "real" and "fake" examples. For example, in a GAN for image generation, the generator produces an image from a random seed. The goal of the generator is to learn the natural distribution of image features and produce images that appear to be sampled from this natural distribution, when in reality they are artificial. The goal of the discriminator is to classify images as either "real" or "fake." The generator works to fool the discriminator, while the discriminator works to recognize the discriminator's output, so GAN training is essentially a two-player minimax game in which the two networks have competing goals [5].

We propose a GAN architecture, where the generator is the HRNN, and the discriminator is a feed-forward CNN. The input to the discriminator is the waveform produced by the HRNN (i.e., a sequence of quantized samples), and the output of the discriminator is the probability that the input is real (i.e, an output of "1" means the input is definitely real, and "0" means the input is definitely fake). The discriminator CNN is implemented as a series of downsampling blocks. The network essentially encodes the input as feature vectors and then classifies them as real or fake.

The adversarial loss is defined as the mean squared-difference between 1.0 and the output of the discriminator. If $\boldsymbol{y}$ is the output of the HRNN, and $D(\boldsymbol{y})$ is the output of the discriminator, then the adversarial loss is defined as:

$$\text{adversarial loss } = (D(\boldsymbol{y}) - 1.0)^2 \tag{10}$$

Because of the differentiability of the regression-based HRNN, $\boldsymbol{y}$ is in fact differentiable, and so is the discriminator, D. Therefore, $D(\boldsymbol{y})$ is differentiable end-to-end across all parameters in the HRNN and the discriminator CNN.

Spectral loss. We may also define "realistic" audio in the frequency domain. That is, perceptually realistic audio should result in a visually realistic spectrogram. We propose an alternative perceptual loss called a "spectral" loss, which is defined as the mean squared difference between the ground-truth log mel-spectrogram and the predicted log mel-spectrogram.

We compute the log mel-spectrogram of an audio signal by first performing the short-time Fourier transform (STFT) of the signal. We take the absolute value of the STFT to compute the magnitude spectrograms. We then convert these to mel-spectrograms using a linear to-mel weight matrix (provided in the TensorFlow contrib package). The log mel-spectrogram is the base-10 logarithm of the mel-spectrogram.

We define the spectral loss as the mean squared difference between the log mel-spectrogram (LMS) of the predicted wideband signal and the LMS of the ground-truth narrowband signal:

$$\text{spectral loss } = (\text{LMS}(\boldsymbol{y}) - \text{LMS}(\text{ground truth}))^2 \tag{11}$$

Both the adversarial loss and the spectral loss are meant to be an additional loss during training. We use a simple L1 loss (the absolute value of the difference between the predicted output and the ground-truth output) for the regression-based output of the HRNN. The total loss to minimize during training is:

$$\text{loss } = \text{L1 } + \text{ perceptual loss} \tag{12}$$

6. Methods

6.1. Dataset

For training and testing we use the VCTK Corpus [18], which consists of speech recordings of 109 different native English speakers. This dataset contains over 18 gigabytes of data, and each

recording is sampled at 48 kHz.

To preprocess the data, we downsampled all recordings from 48 kHz to 16 kHz to form the ground-truth wideband recordings, using an interpolation filter implemented in the Librosa Python package. We then downsampled all recordings from 48 kHz to 8 kHz to form the input narrowband recordings. To ensure that the input sample length matched the output sample length, we upsampled the narrowband recordings to 16 kHz without restoring the high-frequency components. Figure 3 is a diagram of the input pipeline.

We split the dataset into 80/20 train/validation sets. We ran experiments on various scales, including the entire dataset, 2/3 of the dataset, and one of the speakers.

6.2. Implementation

Our model definitions and training and evaluation scripts, implemented in Python and TensorFlow, are available at https://github.com/berthyf96/audio_sr. Much of the code was modified from the Unisound implementation of SampleRNN [16]. The implementation of the discriminator network is borrowed from the work of Pascual et al., who design a GAN for speech enhancement [13].

6.3. Training

We experiment with various hyperparameters for training the model. We first sought to find the right model size for the HRNN. Model size is defined as the dimensionality of the RNN (the number of values in the hidden state vector of each RNN). 1024 dimensions is relatively large and might cause overfitting, so we ran experiments with 1024, 512, and 256 dimensions. We found that we were able to find a good stopping point for training the 1024-dimensional model before there was a deviation between the training and validation loss curves. Given that the largest model does not seem to overfit the training data, we selected 1024 as the model size.

There are two RNN types that result in similar performance. Long short-term memory (LSTM) networks [4] and gated recurrent units (GRU) [1] both feature gate functions to determine which information from previous states to pass on to the next state. Experiments have shown that the two

RNN types have comparable performance [2].

We use Adam optimization, which adapts the learning rate as training progresses [7]. Backpropagation through time (BPTT) is the backpropagation method for RNNs. We use truncated BPTT, updating the HRNN weights every 512 time steps. Note that because each RNN state depends on a previous time step, we cannot predict any samples for the first eight input samples (the first input sequence of the Tier 3 RNN). Therefore, the input sequence length during training is $512 + 8 = 520$ samples.

We mostly trained the baseline HRNN model (without perceptual loss) using the following parameter settings:

Parameter	**Value**
RNN dimensionality	1024
RNN type	LSTM
Optimizer	Adam
Learning rate	0.0001
Truncated BPTT sequence length	520
Sample rate	16000 samples/sec

Table 1: Training hyperparameters.

7. Results

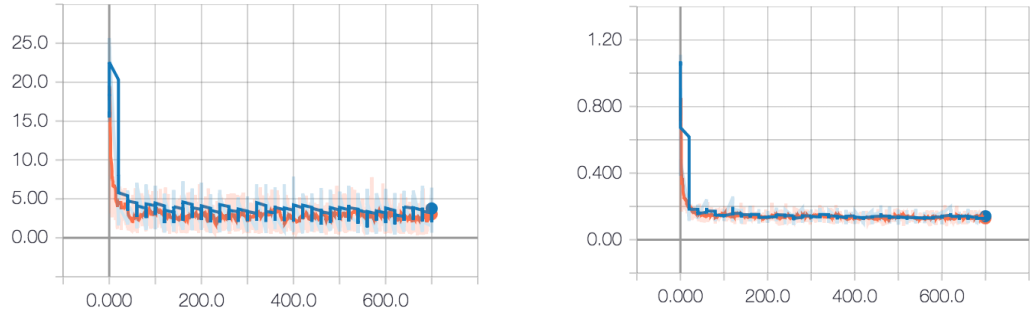
We evaluate the HRNN baseline, HRNN + Spectral Loss (HRNN+Spec), and HRNN + Adversarial Loss (HRNN+GAN) on single-speaker data, specifically speaker "p225" in the VCTK Corpus. The training set consists of 223 WAV files, each at least three seconds long, and the validation set consists of 8 WAV files, each about three seconds long. Note that when training with a sequence length of 520 samples, this means at least 90 training examples per audio file. By evaluating on a single voice, we can better understand the upper bound on the performance of an HRNN on this task, since a multi-speaker HRNN model can perform at most as well as a specialized single-speaker HRNN.

We first compare the training results of HRNN baseline, HRNN+Spec, and HRNN+GAN. Then we provide qualitative and quantitative comparisons of model performance.

7.1. Baseline vs. Spectral Loss vs. Adversarial Loss

We found that HRNN baseline and HRNN+Spec converged fairly quickly during training. We trained both these models for 700 steps with a batch size of 32, which amounts to about 100 epochs.

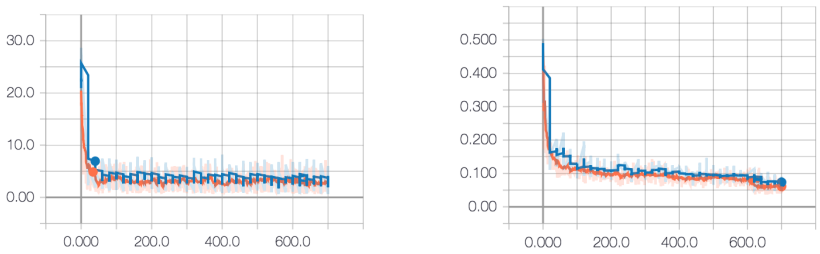
By plotting the test loss and the validation loss during training, we can check for overfitting. We find that even with a large model size of 1024 dimensions, the validation loss curve does not diverge from the train loss curve. Figures 5 and 6 show the loss curves of the baseline and HRNN+Spec, respectively (all loss curves are generated using TensorBoard).



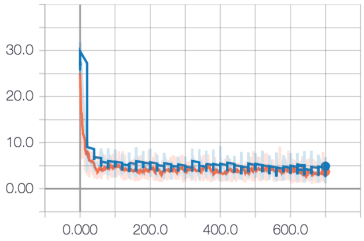
(a) L1 loss (b) Spectral loss

Figure 5: Baseline L1 loss and spectral loss (not part of the loss function) curves. Training loss in orange, validation loss in blue. The baseline model naturally lowers spectral loss without an explicit perceptual loss.

The challenge of training a GAN is reflected in the HRNN+GAN training results. As the loss curves in Figure 7 show, the discriminator learns more quickly. We trained the generator with a learning rate of 0.0001 (the same as HRNN baseline and HRNN+Spec) and the discriminator with a learning rate of 0.00001 to slow down the convergence rate of the discriminator. We also allowed the generator to train for 100 steps before updating the parameters of the discriminator for every 30 steps that the generator trained. It is encouraging that both the adversarial loss and discriminator loss appear to gradually become lower, although at a very slow rate. This suggests that more and/or better training is needed for stronger results. We leave improvement of the training process for future work.

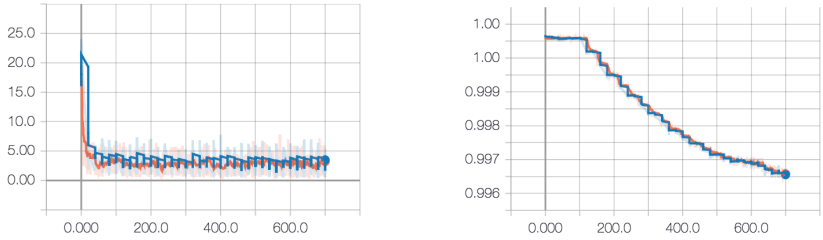


(a) L1 loss (b) Spectral loss

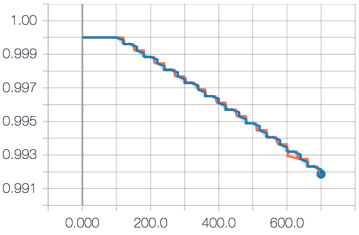


(c) Total loss

Figure 6: HRNN+Spec loss curves. Training loss in orange, validation loss is blue. HRNN+Spec optimizes spectral loss better than baseline.



(a) L1 loss (b) Adversarial loss



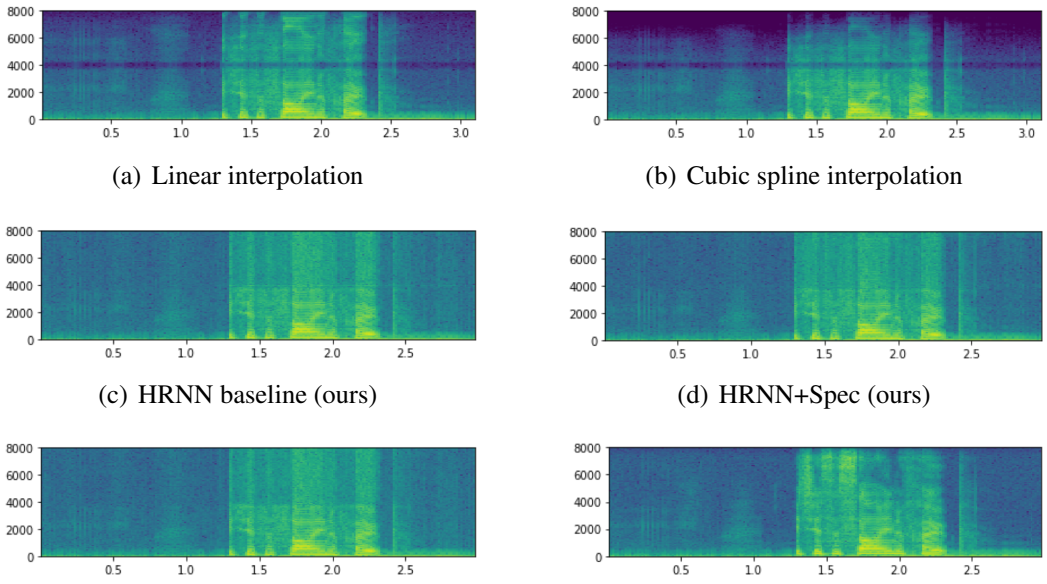
(c) Discriminator loss

Figure 7: HRNN+GAN loss curves. Training loss in orange, validation loss in blue. Even with a 100-step head start, the generator struggles to minimize adversarial loss, as the discriminator learns more quickly.

7.2. Qualitative Results

We find that HRNN baseline, HRNN+Spec, and HRNN+GAN clearly restore high-frequency content to low-resolution audio. The output audio sounds brighter, indicating a widened frequency spectrum. Figure 8 compares the output spectrograms of our methods and interpolation methods and demonstrates that our methods produce much more visually convincing results.

For the qualitative evaluations described here, we used a three-second speech recording from the single-speaker validation set as the input. To generate the output WAV file, since all models were trained with a sequence length of 520 samples, we can either instantiate a model with a longer sequence length of 48008 samples or stitch together output patches of 512 samples using crossfade. We found that the latter method resulted in slightly less noisy output, since the model drifts away from the ground truth as it continues for more time steps. The output examples included in this paper were generated using the crossfade method. We can obtain even better results by improving this method of stitching together 512-long patches. The code for the waveform and spectrogram visualizations is courtesy of Zeyu Jin.



(a) Linear interpolation

(b) Cubic spline interpolation

(c) HRNN baseline (ours)

(d) HRNN+Spec (ours)

(e) HRNN+GAN (ours)

(f) Ground truth

Figure 8: Specotram output comparisons between interpolation methods and our methods. Interpolation methods underestimate the energy at the highest bands and leave a clear low-energy line between lowband and highband. Our methods cover the entire frequency spectrum.

Baseline. Figure 9 shows an example of the input and output of an HRNN baseline trained for 700 steps. The output waveform is more detailed than the input, and the spectrogram of the output shows that the energy at the highband frequencies is restored. While it is difficult to tell of any difference between the ground-truth waveform and the output waveform, there is a visible difference between the ground-truth spectrogram and output spectrogram. The output spectrogram has a clear line between the lowband and highband frequencies. The energy in the highband appears oversmoothed, which is to be expected due to the oversmoothing tendency of RNNs.

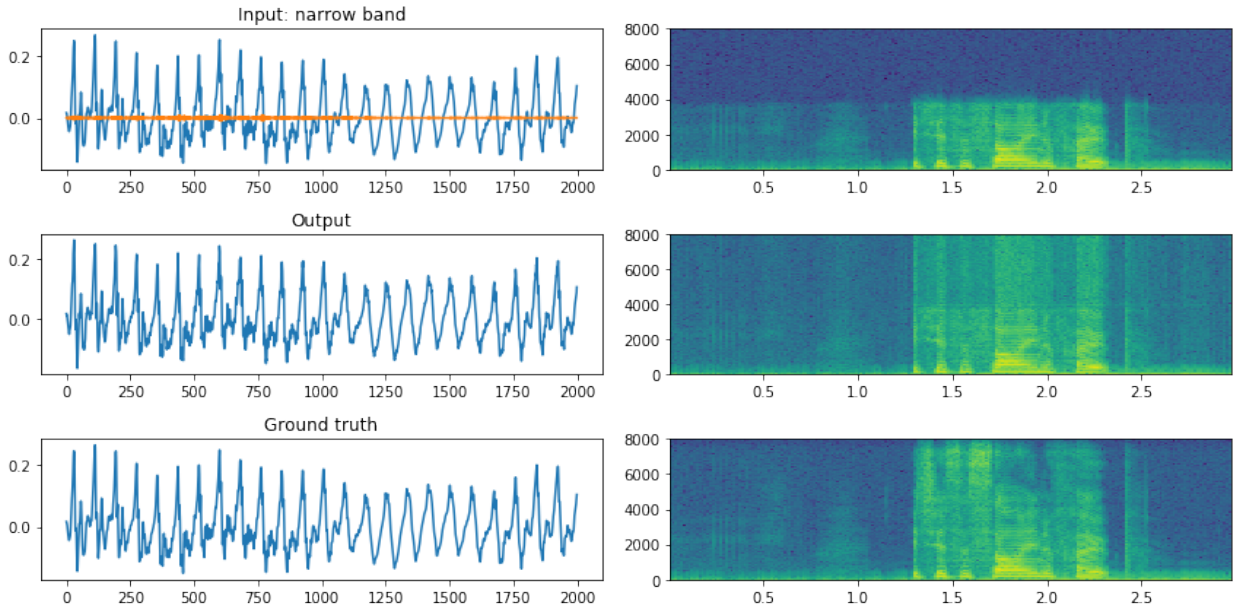


Figure 9: Baseline HRNN output waveform and spectrogram compared to input and ground truth. The orange line plots the difference between the narrowband input and wideband ground truth.

HRNN+Spec. Figure 10 shows an example output of HRNN+Spec. The difference between the lowband and highband frequencies is slightly smoother than with the baseline. Figures 12 and 13 compare the abilities of HRNN baseline and HRNN+Spec to smooth the boundary between the lowband and highband frequency spectra. This model also performs better with silence; the baseline over-estimates the highband energies during silence, whereas HRNN+Spec models silence more closely to the ground truth. In the voiced parts, however, there is still oversmoothing. It turns out that estimating the strength of overtones in a phoneme is difficult, so the model seems to conservatively estimate uniform energy throughout the highband frequency range.

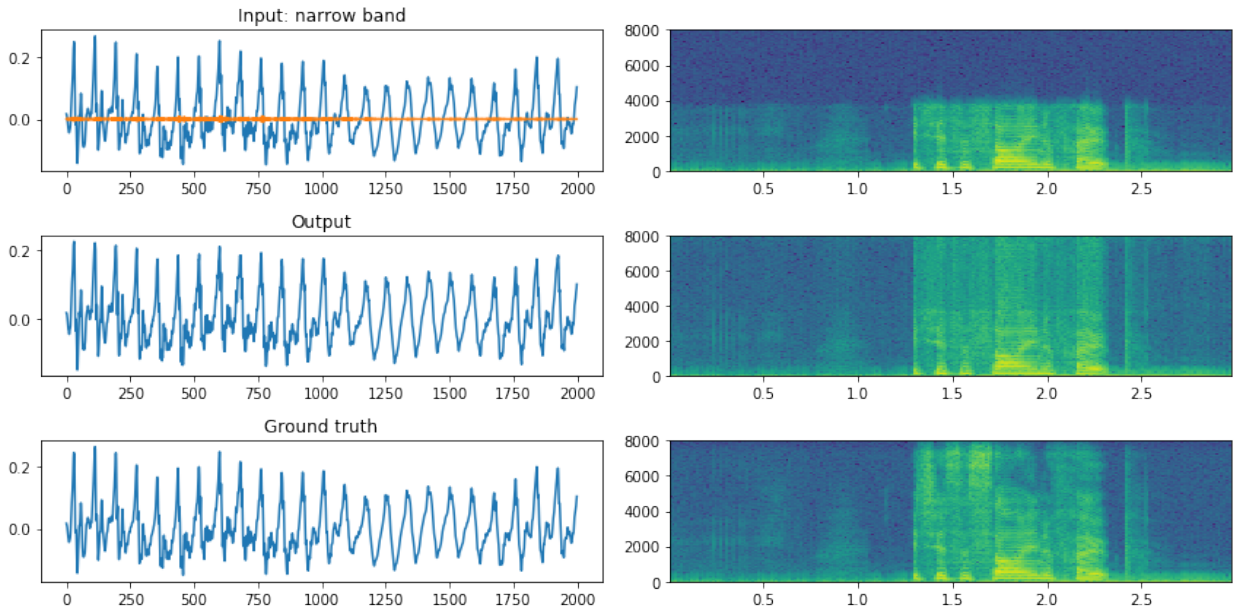


Figure 10: HRNN+Spec output waveform and spectrogram compared to input and ground truth.

HRNN+GAN. Despite the difficulty of minimizing adversarial loss, HRNN+GAN achieves results comparable to HRNN+Spec and better than HRNN baseline. As shown in Figure 11, HRNN+GAN does not oversmooth the upper bands as much as HRNN+Spec, but it also inverts the energies at certain parts of the speech. For example, about halfway through the example recording in Figure 11, HRNN+GAN overestimates the upper band energy at the beginning of syllables but then underestimates the upper band energy during the voiced parts. Because of less oversmoothing, the spectrogram produced by HRNN+GAN indeed seems more visually realistic, but comparing it to the ground truth, we see that the model has missed some of the original high frequencies. There is certainly room for improvement on the adversarial loss.

7.3. Quantitative Results

We compared the performance of HRNN baseline, HRNN+Spec, and HRNN+GAN with the performance of two non-learning based methods: linear interpolation and cubic spline interpolation. We used the metrics of mean absolute difference from the ground truth and mean log spectral distance (LSD) to measure accuracy. We evaluated on the single-speaker validation set. Tables 2 and 3 compare these metrics across methods. Note that the absolute difference is much lower than in training because we evaluate on mu-law decoded sample values between -1.0 and 1.0, whereas

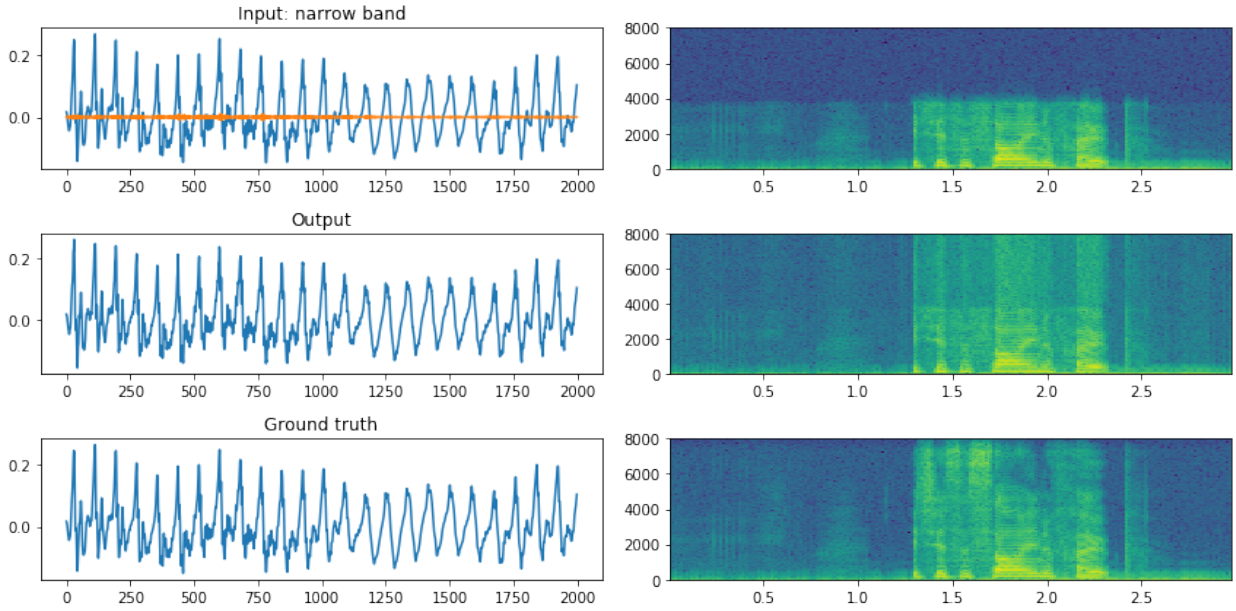
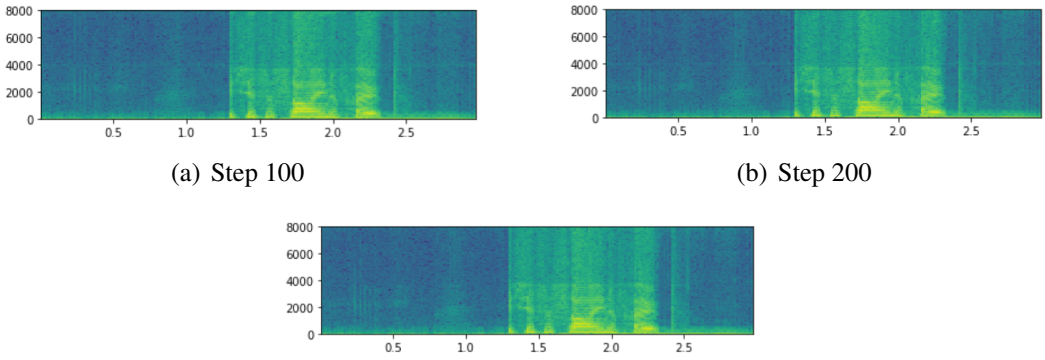


Figure 11: HRNN+GAN output waveform and spectrogram compared to input and ground truth. The output overestimates energy but appears to have close to the correct proportional energies.



(a) Step 100

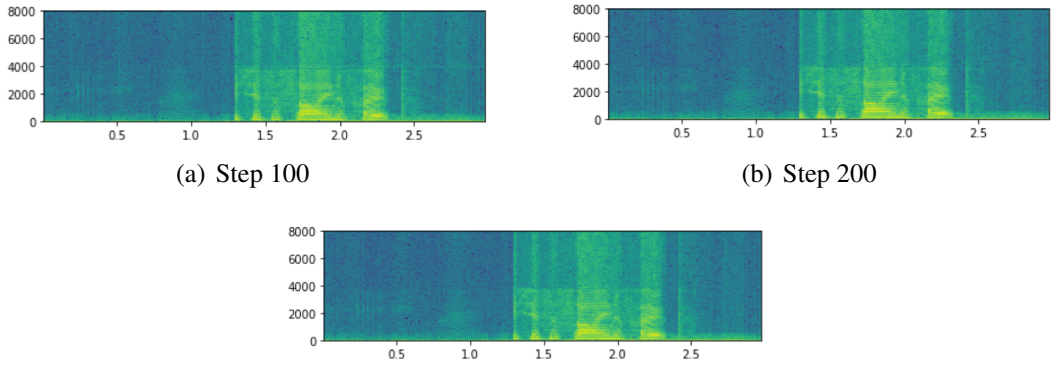
(b) Step 200

(c) Step 300

Figure 12: HRNN+Spec spectrogram output progress in first 300 steps of training. At steps 100 and 200, there is a clear line between the lowband and highband frequencies. At step 300, the line begins to disappear as the model smooths the transition from lowband to highband.

during training the loss is calculated from the mu-law encoded quantized values between 0 and 255.

We found that our methods vastly outperform the interpolation baselines by log spectral distance, indicating that our approach produces more perceptually realistic results. All the models perform similarly by absolute difference. Importantly, the absolute difference between the output waveform and the ground-truth waveform does not always reflect their perceptual similarity. In Figure 8, we see that our methods produce much more visually convincing spectrograms than interpolation methods. For this reason, we believe LSD is a better metric of perceptual quality.



(a) Step 100

(b) Step 200

(c) Step 300

Figure 13: Baseline HRNN spectrogram output progress in first 300 steps of training. Compared to HRNN+Spec, the baseline model produces a more noticeable energy difference between the lower half of the spectrum and the upper half of the spectrum.

HRNN+GAN improves on the baseline by both absolute difference and LSD. This means that the adversarial loss pushes the network to predict more accurate waveform samples that also result in more realistic frequency content. HRNN+Spec outperforms all other methods by spectral distance, which proves the usefulness of an explicit spectral loss.

Model	**Loss**
Linear interpolation	0.00188
Cubic spline interpolation	0.00239
HRNN baseline (ours)	0.00249
HRNN+Spec (ours)	0.00319
HRNN+GAN (ours)	0.00247

Table 2: Mean absolute difference.

Model	**Loss**
Linear interpolation	0.24256
Cubic spline interpolation	0.71407
HRNN baseline (ours)	0.16217
HRNN+Spec (ours)	0.09444
HRNN+GAN (ours)	0.15770

Table 3: Mean LSD.

8. Conclusion

Our proposed HRNN architecture and perceptual losses achieve impressive performance on the audio super-resolution task. The baseline produces acceptable hearing results, and the addition of a perceptual loss encourages even more perceptually convincing output. Specifically, a spectral loss minimizes the spectral distance between output and ground truth, and an adversarial loss encourages perceptually realistic output regardless of ground truth. Since an adversarial loss improves performance at the waveform level, and a spectral loss improves performance at the spectrogram level, the best method for super-resolution models audio in both the time and frequency domains. We believe that our proposals for a perceptual loss may improve any audio generation model, recurrent or feed-forward.

While investigating the properties of RNNs, we found that our methods were susceptible to some of the common drawbacks of RNNs. We noticed overly smooth results and ran into problems training networks with sequence lengths greater than 520 samples. We believe that more sophisticated perceptual losses and training schema, as well as a better method for generating longer sequences, can alleviate both these problems.

We conclude that we were successful in developing a strong baseline HRNN method and improving the baseline with a perceptual loss. We believe the addition of a perceptual loss not only improves performance, but also adds explainability and transparency, as it clearly defines the goal of the model.

9. Future Work

Future work involves improving the HRNN+GAN training scheme by better balancing the generator-discriminator convergence rates and more explicitly conditioning the adversarial loss on the ground truth. There is also a need for a better method for generating longer audio sequences. This might involve crafting a good "patching" method to stitch together short sequences into a long waveform. This would make the model not end-to-end, but it might allow the HRNN to train on even smaller sequences, which would mitigate oversmoothing. Another solution is to allow the HRNN to train

on longer sequences. The length of a phoneme is at least 2000 samples, but a sequence length of longer than 2000 samples causes memory issues during training, and most RNNs cannot accurately model time sequences for that many time steps. Improving the HRNN+GAN training scheme and the method for generating longer sequences would significantly improve our proposed approach.

Our proposed model can be easily extended to multi-speaker data. While we ran many experiments on the entire VCTK dataset, extending the model to many voices requires further work.

We encourage the exploration of other perceptual losses. As discussed earlier in the paper, there are various ways of approaching audio super-resolution, either in the time domain or in the frequency domain. The best perceptual loss reflects the actual listening experience and encourages naturally sounding audio.

Acknowledgements

Thank you to my advisor, Adam Finkelstein, for his valuable support throughout this project. Thank you also to Zeyu Jin, Jiaqi Su, and Riley Simmons-Edler, who were always willing to answer questions (no matter how silly they were) and share their expertise. I am grateful to the Princeton University Computer Science Department and School of Engineering and Applied Science for providing academic, financial, and moral support.

Honor Code

I pledge my honor that this paper represents my own work in accordance with University regulations.

References

[1] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," in *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8), 2014*, 2014.

[2] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv e-prints*, vol. abs/1412.3555, 2014, presented at the Deep Learning workshop at NIPS2014. [Online]. Available: https://arxiv.org/abs/1412.3555

[3] Y. Gu, Z.-H. Ling, and L.-R. Dai, "Speech bandwidth extension using bottleneck features and deep recurrent neural networks," in *INTERSPEECH*, 2016.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[5] M. M. B. X. D. W.-F. S. O. A. C. Y. B. Ian J. Goodfellow, Jean Pouget-Abadie, "Generative adversarial networks," in *NIPS*, 2014.

[6] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," 2016.

[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[8] V. Kuleshov, S. Z. Enam, and S. Ermon, "Audio super-resolution using neural nets," in *ICLR (Workshop Track)*, 2017.

[9] C. Ledig, L. Theis, F. Huszár, 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," in *CVPR*, 2017.

[10] K. Li, Z. Huang, Y. Xu, and C.-H. Lee, "Dnn-based speech bandwidth expansion and its application to adding high-frequency missing features for automatic speech recognition of narrowband speech," in *INTERSPEECH*, 2015.

[11] Z.-H. Ling, Y. Ai, Y. Gu, and L.-R. Dai, "Waveform modeling and generation using hierarchical recurrent neural networks for speech bandwidth extension," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 26, no. 5, pp. 883–894, May 2018. [Online]. Available: https://doi.org/10.1109/TASLP.2018.2798811

[12] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio, "Samplernn: An unconditional end-to-end neural audio generation model," in *ICLR*, 2017.

[13] S. Pascual, A. Bonafonte, and J. Sera, "Segan: Speech enhancement generative adversarial network," in *INTERSPEECH*, 2017.

[14] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerry-Ryan, R. A. Saurous, Y. Agiomyrgiannakis, and Y. Wu, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *Arxiv*, 2017. [Online]. Available: https://arxiv.org/abs/1712.05884

[15] S. B. Unadkat, M. M. Ciocoiu, and L. R. Medsker, *Recurrent Neural Networks: Design and Applications*. Washington, D.C.: CRC Press, 2001.

[16] Unisound, "Samplernn: Tensorflow implementation of samplernn," https://github.com/Unisound/SampleRNN, 2017.

[17] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *Arxiv*, 2016. [Online]. Available: https://arxiv.org/abs/1609.03499

[18] J. Yamagishi, "Cstr vctk corpus," http://homepages.inf.ed.ac.uk/jyamagis/page3/page58/page58.html, 2010.