ASRWGAN: Wasserstein Generative Adversarial Network for Audio Super Resolution

Jonathan Gomes-Selman, Arjun Sawhney, Woody Wang
Department of Computer Science
Stanford University
jgs8@stanford.edu, sawhneya@stanford.edu, wwang153@stanford.edu

Abstract

This paper proposes using a Wasserstein Generative Adversarial Network to enhance the performance of an existing neural network trained to perform audio super resolution. Inspired by SRGAN [3], we utilize a pre-trained version of ASRNet, as described by Kuleshov et al. [2], as a generator with a fully convolutional discriminator model. We use an adapted loss function for the generator, summing a content loss (MSE between the generator output and corresponding true audio signal) with the traditional adversarial loss. Our results show that our model performs considerably better than a bicubic interpolation baseline in both signal-to-noise ratio (SNR) and log spectral distance (LSD). In comparison to ASRNet, our model shows stronger performance on the LSD metric and reduced SNR due to its attempt to more bravely reconstruct higher frequencies of the low resolution signals. Both ASRNet and our model achieve near identical performance on MUSHRA tests, which incorporate human perception of the clarity of the produced audio signal, and both significantly outperform the baseline.

1 Introduction

With the rise of personal assistant systems and audio data, auditory inputs toward technological devices are becoming more and more prevalent; however, given the coarsity and variability of sounds and subtle differences in recording devices, systems that take audio as input often have to deal with poor quality audio and at times must re-confirm or repeatedly ask the same questions to interpret the input. As such, a network that could take poor quality audio as input and enhance, or super-resolve, it without requiring confirmation or repetition from the user could improve the experience of personal assistants and other technologies that use audio data to inform actions.

Given this motivation, we propose to improve an existing model which performs bandwidth extension, a specific form of audio super resolution, by reconstructing high-quality audio from a low-quality, down-sampled version as described by [2]. Taking into account the success of SRGAN [3] which utilizes a Generative Adversarial Network (GAN) to improve an existing model for Image Super Resolution, we propose a modified Wasserstein GAN architecture to enhance a model for audio super-resolution, ASRNet, introduced by [2]. By coupling a modified version of ASRNet as the generator with a deep convolutional discriminator, our ASRWGAN's results show promise for using GANs to augment current methods of audio super resolution.

2 Related work

Our work draws inspiration from a variety of deep learning approaches both for audio and non-audio related applications. Fundamentally, as aforementioned, the primary goal of our project is to improve

CS230: Deep Learning, Spring 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

on the performance of an existing deep residual network (ASRNet) for audio super-resolution. Proposed by [2], ASRNet draws inspiration from previous research on image super-resolution and is modeled as a deep convolutional neural network with residual skip connections. ASRNet has been shown to greatly outperform traditional interpolation techniques and provides a promising, real-time network architecture for bandwidth extension (audio super resolution).

We propose a model to further improve the performance of ASRNet by incorporating the benefits of Generative Adversarial Networks. As noted by [2], the task of audio super-resolution greatly mirrors that of image super-resolution. Therefore, our proposed model and methods closely relate to those presented in SRGAN [3], a GAN for image super resolution shown to outperform previous state of the art architectures for super-resolving images at large scaling factors.

In developing our GAN architecture, we also draw inspiration from the implementation of WaveGAN[1]. WaveGAN explores the problem of audio synthesis using fully convolutional architectures (to process audio signals) as opposed to using RNNs which are more closely associated with time-series modelling. WaveGAN's performance demonstrates the potential for applying convolutional models to one dimensional time series data, which is an approach we further pursue here.

3 Dataset and Features

The data we use comes from the CSTR VCTK Corpus provided by the Center for Speech and Technology Research [5]. This dataset includes speech data from 109 native English speakers reciting around 400 English sentences each, although we are only training on data from a single speaker for the sake of efficiency and due to compute time limitations. As an aside, we note that in future applications, we can foresee audio super-resolution models specifically trained for individual speakers in services such as Skype or Alexa. The data is in the format of WAV files, which we convert to a numpy array using Python's librosa library with a fixed sampling rate of 16,000. We represent an audio signal from the WAV files as a function $f(t) : [0, T] \rightarrow \mathbb{R}$, where $f(t)$ is the amplitude at t and T is the length of the signal. To process the continuous signal as an input, we must discretize $f(t)$ into a vector $x(t) : [\frac{1}{R}, \frac{2}{R}, ..., \frac{RT}{R}]$, where R is the sampling rate of the input in Hz. For this audio super-resolution task, we consider R to be the resolution of the input x.

In order to standardize input length, we sample half second patches from recordings in the dataset, resulting in vectors of shape (8192, 1) after preprocessing. We then shuffle the vectors randomly and perform the following train/val split:

Train: 3328 examples, Validation: 500 examples

We pre-process each high resolution example by using a Chebyshev low-pass filter to decimate the initial signal into a low resolution equivalent, which we provide as input for our generator network. As a baseline for reconstruction we apply bi-cubic interpolation.

4 Methods and Final Model

Shown in Figure 1, our proposed model draws inspiration from three main sources: SRGAN, WaveGAN, and ASRNet [1][2][3]. Our general approach follow closely the methodology proposed in SRGAN, which includes the use of a pre-trained generator network (ASRNet) to avoid undesired local minima and the incorporation of a content loss component into the generator loss to enhance performance by providing domain knowledge about the actual task at hand (super resolution). For the individual components of the GAN architecture (generator and discriminator) we use modified versions of ASRNet and the WaveGAN discriminator respectively. We choose the WaveGan discriminator specifically because of its demonstrated performance on audio time series data, which largely relates to the success of long 1D convolutional filters in capturing the periodic nature of sound.

During our architecture search we focused mainly on two types of GAN, namely the Vanilla GAN and a modified Wasserstein GAN (WGAN) [4]. Our initial implementation was a Vanilla GAN employing traditional GAN training techniques, such as the use of the non-saturating cost function, and the Leaky RELU non-linearity (as described in CS230 lecture) among others. However, this model exhibited mode collapse, exploding gradients, and varying losses whilst training and as such we turned to the WGAN for greater training stability.

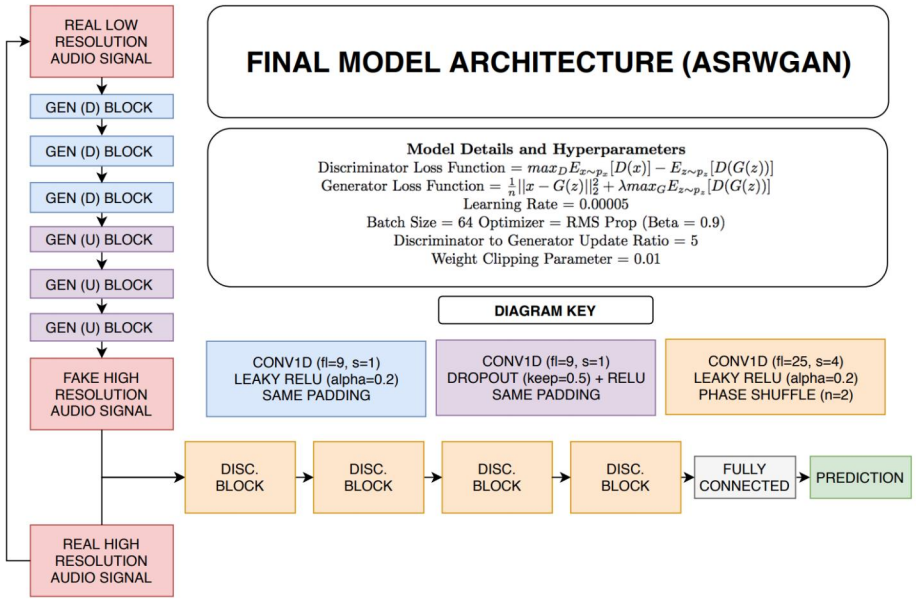

Figure 1: Final ASRWGAN Architecture with tuned hyperparameters

The WGAN adapts the Vanilla GAN by re-defining the loss function, incorporating weight clipping, reducing the learning rate, and using a non-momentum based optimizer (RMS Prop). As mentioned in [4], these changes serve to allow more reliable gradients to back propagate to the generator, whilst keeping parameter values small in order to prevent issues like mode collapse.

Modified Loss functions:

$$DiscriminatorLoss = max_D E_{x\sim p_x}[D(x)] - E_{z\sim p_z}[D(G(z))] \tag{1}$$

Following the WGAN training algorithm outlined in [4], the discriminator is no longer trained to identify real and predicted examples, but now trains to compute the Wasserstein distance.

$$GeneratorLoss = \frac{1}{n}||x - G(z)||_2^2 + \lambda max_G E_{z\sim p_z}[D(G(z))] \tag{2}$$

We modify the generator loss proposed in [4] to incorporate a content loss in addition to the traditional adversarial loss due to the success of a similar approach in [3]. We use MSE (mean squared error) between predicted and true examples to provide domain knowledge about the actual task goal (super resolution) and an additional hyper-parameter λ to balance the content and adversarial loss. Specifically, we include λ to control our model's emphasis on optimizing for the content loss.

5 Results and Discussion

Metrics: We use signal to noise ratio and log spectral distance as suggested metrics per Kuleshov et al. [2]. Given a target signal y and a reconstruction x, the SNR and LSD are defined in equations (3) and (4) respectively, where X and $\hat{X}$ are the log-spectral power magnitudes of x and y, which are defined as $X = log|S|^2$, where S is the short-time Fourier transform of the signal, and l, k are the index frames and frequencies, respectively.

$$SNR(x, y) = 10log\frac{||y||_2^2}{||x - y||_2^2} \tag{3}$$

$$LSD(x, y) = \frac{1}{L}\sum_{l=1}^{L}\sqrt{\frac{1}{K}\sum_{k=1}^{K}(X(l, k) - \hat{X}(l, k))^2} \tag{4}$$

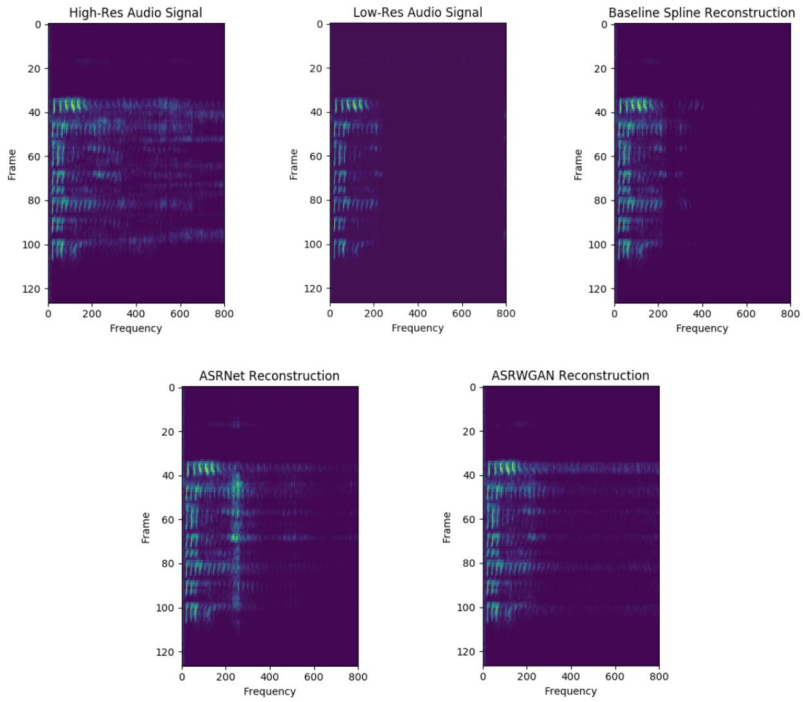

Figure 2: Example spectograms from various reconstruction methods

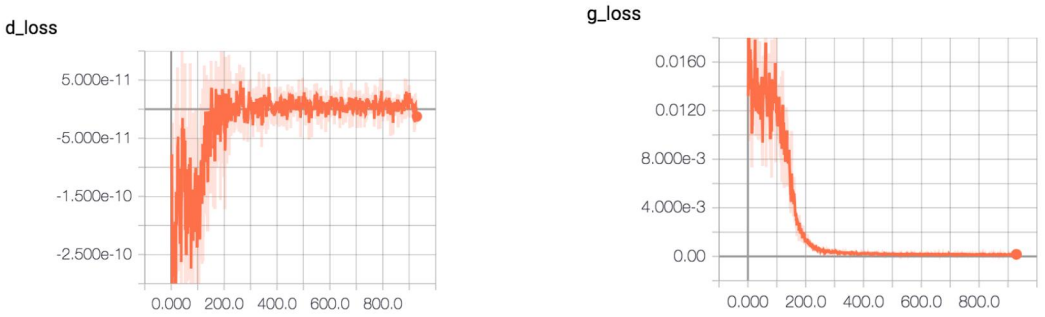

Figure 3: Discriminator and generator loss curves with the respective loss on the y-axis and mini-batch iteration number on the x-axis

Evaluation: Seen in Figure 2., compared to the baseline bicubic interpolation, or spline, reconstruction, our ASRWGAN reconstruction shows significant improvement in resolving higher frequencies. Relative to the ASRNet, our ASRWGAN still recovers more of the higher frequency, but displays some extraneous noise that is not present in the original high-res signal, which can be addressed by preserving more of the pre-trained generator model.

From the objective metrics in Table 1 and 2, we see a slight decrease of approximately 1-2 dB in SNR, but an improvement (reduction) in LSD. The LSD metric rewards resolving higher frequencies more than the SNR metric [2]. After closer examination of saved weights over epochs 1-5, the SNR drops significantly to ~ 5, which suggests a diminishing benefit of leveraging a pre-trained network, likely due to a performance gap between our discriminator and generator; however, we see a quick recovery and improvement upon baseline performance over as few as 40 epochs. Thus, although we seem to introduce more noise compared the the ASRNet generator, we recover more of the high frequencies as a tradeoff.

Objective Metrics	Spline	ASRNet	ASRWGAN
Signal to Noise Ratio	14.8	17.1	15.7
Log Spectral Distance	8.2	3.6	3.3

Table 1: Objective evaluation of audio super-resolution methods at an upscaling ratio of 4

MUSHRA	Sample 1	Sample 2	Sample 3	Average
ASRWGAN	70	61	73	68
ASRNet	67	63	75	68.3
Spline	42	34	36	37.3

Table 2: Average MUSHRA user study scores for each audio sample

After adding weight and gradient clipping and transitioning to a WGAN, we avoid mode collapse and see improved stability in training as shown by our loss curves in Figure 3. We can see the generator loss steadily decreases while the discriminator continues to be updated over successive iterations.

Suggested by the work of Kuleshov et al. [2], we then reaffirmed our objective metrics through asking 10 classically trained musicians to rate the general quality of reconstruction using a MUSHRA (MUltiple Stimuli with Hidden Reference and Anchor) test. We randomly selected three audio samples from the VCTK single speaker dataset, downsampled the samples, and reconstructed the samples using a bicubic spline interpolation, ASRNet, and our ASRWGAN. We then asked each subject to rate each sample on a scale of 0 (terrible) to 100 (perfect). The results of this experiment are seen in Table 2., where we see a significantly higher rating for our ASRWGAN reconstruction compared to the spline, but a less noticeable difference between ASRNet's and ASRWGAN's reconstructions.

6 Conclusion/Future Work

In conclusion, we introduced a new deep architecture for audio super resolution. Our approach is novel in that it combines recent promising work from SRGAN, WaveGan and ASRNet [1][2][3]. Empirical evaluation shows improved performance compared to more traditional methods, especially for the high-frequency components of the audio signal. A human subject evaluation also rated the final results superior compared to traditional methods.

We experimented with both a Vanilla GAN and Wasserstein GAN (WGAN) for our final architecture. We settled on WGAN, which tended to suit our problem best due to the smaller learning rate, modified loss function and addition of weight and gradient clipping which improved training stability.

Our strongest model, entitled ASRWGAN, performs considerably stronger than the traditional bicubic interpolation methods in both signal-to-noise ratio (SNR) and log spectral distance (LSD) whilst showing stronger performance on the LSD metric and reduced SNR when compared to ASRNet. We argue these results are consistent with the observation that our model attempts to reconstruct the highest frequencies of the input audio signal, which is arguably the most challenging part for audio super resolution. The model does introduce some noise in the form of discontinuities to the predicted output signal. Qualitatively, our MUSHRA experiments indicate comparable output signal clarity with ASRNet and far superior clarity than our baseline model.

We hypothesize that the structure of the ASRWGAN, in particular the initial performance gap between the discriminator and generator, causes ASRWGAN to not be able to fully take advantage of the initial pre-trained state of the generator. In light of this, our first action for future work is to make our discriminator more expressive through the introduction of skip connections and residual units and to tune the discriminator-to-generator training ratio. Additionally, we intend to experiment with the loss function on the generator, in particular the content loss, in order to better reflect our eventual goal of optimizing performance for the human ear. Given more compute time, a natural extension would be to train our model on multiple speakers in the VCTK data set and perform a more thorough hyper parameter search for the weight clipping bounds.

7 Contributions

Each team member contributed equally to the project. Jonathan wrote most of the code to create the architecture of our ASRWGAN. Woody worked on writing code to train and test the model on the AWS instance. Arjun worked on preprocessing the dataset and adapting and implementing the GAN architecture based on analysis of existing GAN implementations in the audio and image super-resolution domains. All team members thoroughly read relevant papers to increase group productivity, spent time debugging and refactoring code, and contributed equally in the composition of the poster, milestone and final reports.

8 Acknowledgements

We would like to thank Volodymyr Kuleshov for his support throughout our project, helping us understand his initial model's architecture. We would also like to thank Ahmad Momeni, Brandon Yang and the entire CS230 teaching staff for helping us implement and understand our ASRWGAN.

9 Code

The project code is available at https://github.com/jonathangomesselman/CS230-Project

References

[1] Donahue, Chris et al. Synthesizing Audio with Generative Adversarial Networks in arXiv, 2018.

[2] Kuleshov, Volodymyr et al. Audio Super Resolution with Neural Networks in arXiv (Workshop Track) 2017.

[3] Ledig, Christian et al. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network in arXiv 2017.

[4] Arjovsky, Martin et al. Wasserstein GAN in ICML, 2017.

[5] English Multi-speaker Corpus for CSTR Voice Cloning Toolkit, 2010.

[6] Abadi, Martín et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org, 2015.