Audio Super Resolution - Final Report

Chi Wang, Zheng Yuan and Abhipray Sahoo
{chiwang,zyuan84,abhipray}@stanford.edu

1 Introduction

Generative modeling of audio signals can be useful for the task of super audio resolution. Our task is to reconstruct high-quality audio from a low-quality, down-sampled input containing only a small fraction (15- 50%) of the original samples. This would be very useful in various applications such as voice/music recognition, telephony, compression, text-to speech generation, and other domains.

In signal processing literature, this problem is also called bandwidth extension or upsampling since increasing sample rate leads to generation of high frequency components. Usual techniques like low-pass filtering after zero-filling in-between samples, leads to no new information content in the high frequencies. Our problem is to use learned prior statistics of a typical speech file to predict these high frequencies from the low frequencies of the audio.

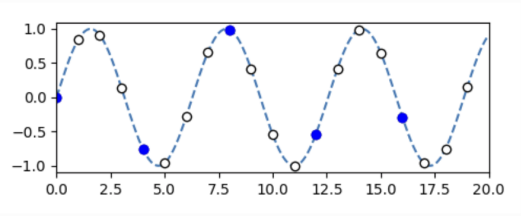

Figure 1: generate new time-domain samples in an audio signal

We saw many success stories of applying neural network on image super resolution, but not so many in audio field. The traditional data augmentation approaches like mathematical interpolation and standard neural network does not work very well. So we want to leverage generative approach to help upsampling from a low resolution audio file with high quality, and we hope to get it work as lightweight fashion, so we could apply it in real-time product.

In this project, we explore three different generative models for bandwidth extension and compare them.

2 Related Work

Our project is heavily influenced by following two approaches: audio super resolution (Volodymyr Kuleshov et al., 2017)[1] and image super-resolution algorithms (Dong et al., 2016) [2], which use customized asymmetric auto-encoding/decoding techniques to interpolate a low-resolution image into a higher-resolution one. We choose it as a baseline approach since it runs efficiently and potentially could generate good results. In order to improve the performance, we designed a more sophisticated upsampling block and replaced the original upsampling block in the decoder.

Since we are also interested in how pure generative algorithm performs on this setting, we explored two purely generative approaches: "WaveGlow: A Flow-based Generative Network for Speech Synthesis" [3] and "sampleRNN: an unconditional neural audio generation model" [4], then compare them with the auto encoder like approach.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

3 Problem Statement

The problem is to reconstruct a low resolution signal $x = \{\frac{x_1}{r_1}, ..., \frac{x_{r_1 t_1}}{r_1}\}$ sampled at a rate r_1, to a high resolution version $y = \{\frac{y_1}{r_2}, ..., \frac{y_{r_2 t_2}}{r_2}\}$ of x that has a sampling rate $r_2 > r_1$. For example, x may be a voice signal recorded at 4 KHz,we try to reconstruct sequence y as a high resolution 16KHz of the original (x) if we choose upsampling rate as 4.

3.1 Dataset

For voice dataset, we use the VCTK dataset (Yamagishi) — which contains 44 hours of data from 108 different speakers, each audio file is nearly 30 seconds long. These are all high quality audios. To get low resolution audio, we down sample them at rate 1/2,1/4 and 1/6, we pair the low resolution down sampled audio with the original audio for training. See the example below.

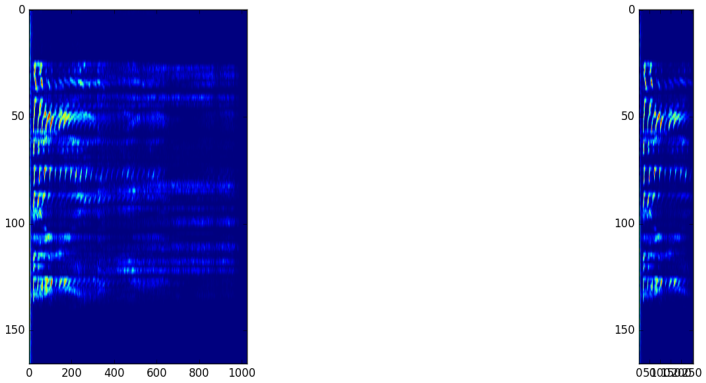

Figure 2: Spectrum of high/low resolution audio files

Dataset download and process code could be found here: VCTK voice data download and process code

The LJSpeech dataset is used to train the WaveGlow model. This dataset contains roughly 24 hours of speech from a single speaker with each clip being between 1-10seconds. The dataset can be found at https://keithito.com/LJ-Speech-Dataset/.

4 Technical Approach

4.1 Approach: Upsample decoder

This is based on the paper audio super resolution (Volodymyr Kuleshov et al., 2017)[1]. Continuing from problem statement section, for low resolution signal x and its reconstructed signal y, we use $r = \frac{r_2}{r_1}$ to denote the upsampling ratio of the two signals, which in our work equals $r = 2, 4, 6$. We thus expect that we could train a model to produce below result:

$$\frac{y_{rt}}{r_2} \approx \frac{x_t}{r_1}, \quad fort = 1, 2, ..., t_1 r_1$$

To recover from the low resolution signal, we want to define a neural network to learn $p(y|x)$ of the heigher-resolution y, condition on its low resolution sequence x. We assume that relationship between the time series x, y follows the equation $y = f_\theta(x) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$ is Gaussian noise, and we could get a $f_\theta(x)$ is close to values sampled from $p(y|x)$, we might also try more complex noise models for ϵ if there is time.

4.1.1 Architecture

We kept the original architecture in audio super resolution (Volodymyr Kuleshov et al.,2017)[1], and add more Conv layer in the upsampling block and more U blocks in decoder, we hope it could capture more wavelet-style features with new added layers.

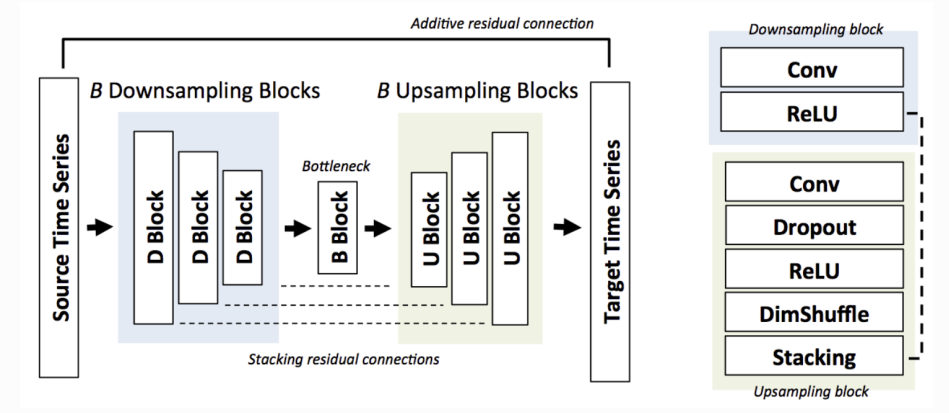

Figure 3: Model Architecture

4.1.2 Loss Function

From above model architecture, it naturally leads to a mean squared error (MSE) objective:

$$loss(Dataset) = \frac{1}{n}\sqrt{\sum_{i=1}^{n} \|y_i - f_\theta(x_i)\|_2^2}$$

Since we have the original audio file associated with reconstructed high resolution audio, this loss function could be calculated easily.

4.2 Approach: WaveGlow

WaveGlow is a normalizing flow model that explicitly models the probability distribution of audio sample sequence x conditioned on a mel-spectrogram representation h of the audio $p(x|h)$. It generates samples by first drawing a hidden random vector z from a standard normal distribution and applying a bijective series of transforms to get audio samples. It is trained by maximizing the log-likelihood of the data. The data in this case is pairs of a limited resolution mel-spectrogram and the original high-resolution audio samples.

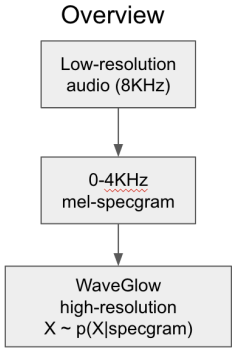

Figure 4: bandwidth extension with WaveGlow

Once WaveGlow's generative model has been trained, we can apply it for the downstream task of super audio resolution by simply conditioning on the mel spectrogram of the low resolution audio and drawing samples from the the standard normal distribution for the hidden latent variable.

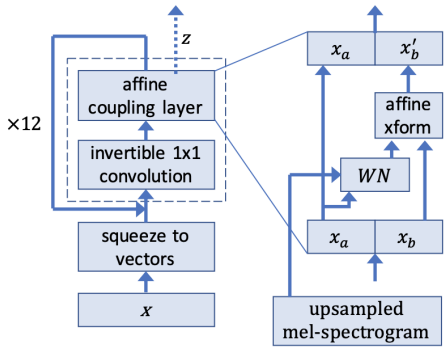

Fig. 1: WaveGlow network

The architecture of the network involves combination of Wavenet-like[5] diluted convolution operations with Glow's[6] multi-scale architecture that includes several steps of flows. Each flow has an invertible 1x1 convolution layer that learns weights for a random rotation permutation matrix, and an affine coupling layer that allows the bijective transformation to be lower-triangular.

The reason we chose WaveGlow as a generative model is because the flow model has fast synthesis compared to other models like WaveNet. While WaveNet has been modified to have faster generation via parallel wavenet[7], it is still challenging to train requiring a two step teacher-student training procedure.

5 Results

As evaluation metric, we use Signal to Noise Ratio (SNR) to compare the upsampled audio with original audio. Given a reference signal y and an approximation x, SNR is defined as:

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

We also use the Log Spectral Density defined as:

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

LSD measures reconstruction quality as a function of the spectrogram.

5.1 Result: Upsample decoder

5.1.1 Training Metric

Average metrics	original architecture in paper	upgraded architecture
LSD	16.98 (dB)	17.3 (dB)
SNR	17.1 (dB)	17.5 (dB)

Table 1: Average SNR and LSD of upsampling reconstruct audio files across all 250 validation samples.

5.1.2 Audio Spectrum Comparison

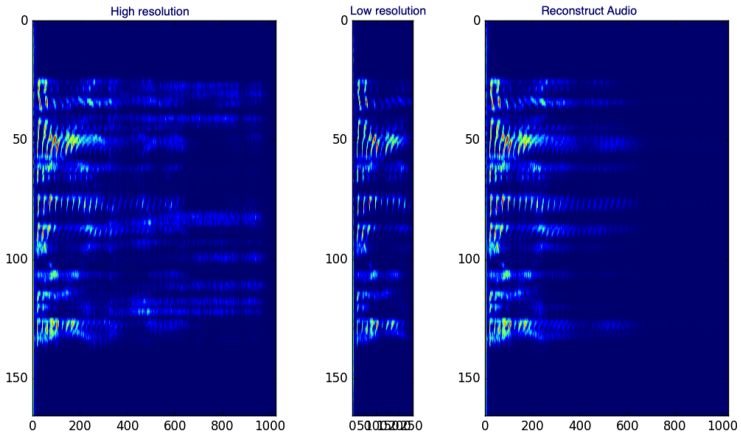

Figure 5: Audio spectrum on original model in paper

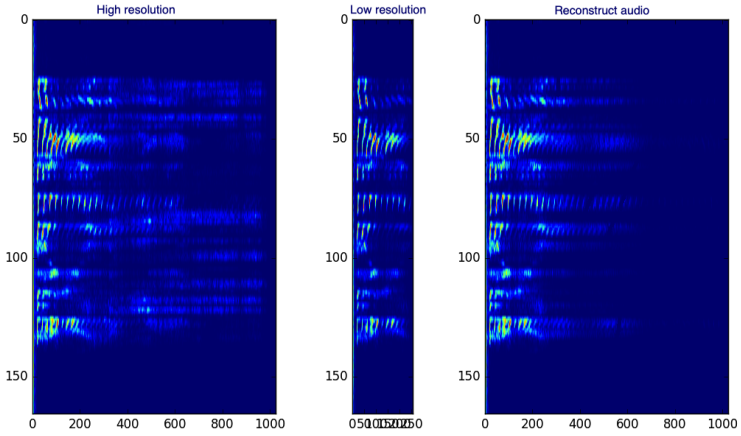

Figure 6: Audio spectrum on enhanced model

5.1.3 Analysis

Baseline: For approach 4.1 - Upsample decoder, we choose the architecture defined in paper [1] as baseline, and trained both upgraded model and original model on VCTK dataset (108 speaker, 44 hours audio) to see if there is any improvement.

Evaluation: During test stage, we run both models on 250 preserved audio files (16KHz) from VCTK dataset, which takes these audio files as high resolution files, downsample to low resolution audios (4KHz) and then reconstruct it back to 16KHz. Finally, We calculate the average SNR and LSD between the high resolution audio and the reconstructed audio files, and compare their numbers. The result is list in Table 1 (see above)

By looking at the LSD and SNR result, we find the model performance is nearly the same between the enhanced model and original model, it seems like adding more convolution layer and connect layer doesn't make dramatic improvement. We want to experiment on different hyper-parameter and different downsampling algorithm as next step.

5.2 Result: WaveGlow

Training: In our experiments, we re-trained the model from scratch on VCTK but did not get intelligible speech generation after a week of training. Instead, for model comparisons we used

pre-trained weights trained on the LJSpeech dataset; the training was done with mel-spectrogram that has 80 mel frequency values between 0-8KHz. The sample rate of audio during training was 22.05KHz.

Baseline: The baseline is Resampy's implementation [8] of an upsampler that uses a high-quality Kaiser window for the low-pass filter in the upsampling step.

Evaluation: During test time, we generate the melspectrogram of a 16Khz audio and upsample via WaveGlow to 22.05Khz audio.

Qualitative analysis: The generated audio is highly intelligible but the SNR is low due to artifacts from the model's biases such as periodic hums (see strong high frequency lines in figure 4). Additionally this method alters the original low frequency information; therefore in the future, we will explore conditioning the WaveGlow network with the original audio samples so only the high-frequency components are generated.

Quantitative analysis: The following example figure and table compares the performance of WaveGlow to the chosen baseline. The table captures the average SNR and LSD measured on ten audio samples. The WaveGlow generated audio performs worse on both the SNR and LSD metrics than the baseline. We note that through subjective listening tests that the worse SNR and LSD numbers do not imply bad listening quality.

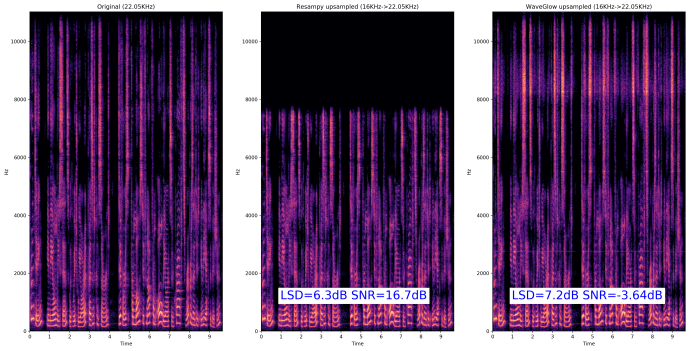

Figure 7: Example upsampling by Resampy and WaveGlow

Average metrics	(dB)
WaveGlow LSD	7.573473
Kaiser LSD	6.437214
WaveGlow SNR	-3.302849
Kaiser SNR	17.047472

Table 2: Average SNR and LSD of upsampled files across ten different samples.

5.3 Upsample decoder vs WaveGlow

Since our upsample decoder only work on integer sampling rate when reconstructing low resolution audio (ex: 4KHz to 16KHz with scale as 4) and the pretrained WaveGlow model only works upsampling audio file from 16KHz to 22.5KHz, it is difficult to do an apple to apple comparison between these two models. Instead, we pick a random audio file from LJSpeech dataset, feed it to both model and compare their SNR / LSD on the results. (See result below in Table 3)

Average metrics	WaveGlow	Upsample Encoder
LSD	7.6 (dB)	10.5 (dB)
SNR	15.6 (dB)	11.7 (dB)

Table 3: SNR and LSD of upsampling reconstruct audio files on 5 audio files in LJSpeach dataset.

From the result, we see WaveGlow model perform better than Upsample encoder, but as we mentioned above, this is not a strong evidence to show WaveGlow is better, since we run Upsample decoder on a dataset it never seen and their upsample rate is different, one is 4KHz to 16KHz, another is 16KHz to 22.5KHz. As a next step, we need to udpate both models to get them work on same upsampling rate to get a fair comparison.

6 Code

Code and analysis scripts for our WaveGlow model can be found at https://github.com/Abhipray/waveglow, for upsample encoder approach, code can be found at https://github.com/wcAlex/audio-super-res-imgres-approach.

7 Conclusion

In this project, we explored three different kinds generative approaches in audio bandwidth extension field. One is to extend auto-encoder architecture in [1] by replacing the decoder part with more sophisticated upsampling block. Second one is WaveGlow [3], which uses normalized flow method and the third one is SampleRNN 8 which we didn't get to the release stage due to time constraints and the complexity of extending it to be conditioned on low resolution audio files.
Based on our experiments, we conclude that our Upsample Encoder method has a little improvement over the original model and WaveGlow seems to perform better qualitatively although its upsampling factor is lower and so is dealing with a simpler problem.

8 Future Work: SampleRNN based approach

We argue that sampleRNN based method should perform better than our current method and two baselines. The intuition is that under conditional sampleRNN model, each high resolution sample is conditioned on both its previous samples and also its low resolution version. Thus, the sampleRNN is designed to learn not only the extrapolation from low resolution to high resolution but also the transition among high solution samples along time. The other intuition is that the original samples could be forced to be the intact unlike the WaveGlow model.

8.1 From sampleRNN to conditional sampleRNN

Vanilla sampleRNN (Fig. 8) is an autoregressive model to generate audio samples. In its vanilla version, each sample is generated based on the samples that come before it. The building block of SampleRNN is a hierarchy of tiers, which in turn consist of multiple layer of recurrent neural network and also multilayer perceptrons. This model is designed to capture the variations on sequences with long time span. Fig. 8 is an unrolled version of sampleRNN to output $p(x_{i+4}|x_{<i+4})$ ($\underline{Xi+4}$ in Fig. 8), $p(x_{i+5}|x_{<i+5})$, ..., $p(x_{i+11}|x_{<i+11})$, as the sample level. As you can see in the model, each of this output samples only depends on the samples before them, thus conforming to the autoregressive ordering.

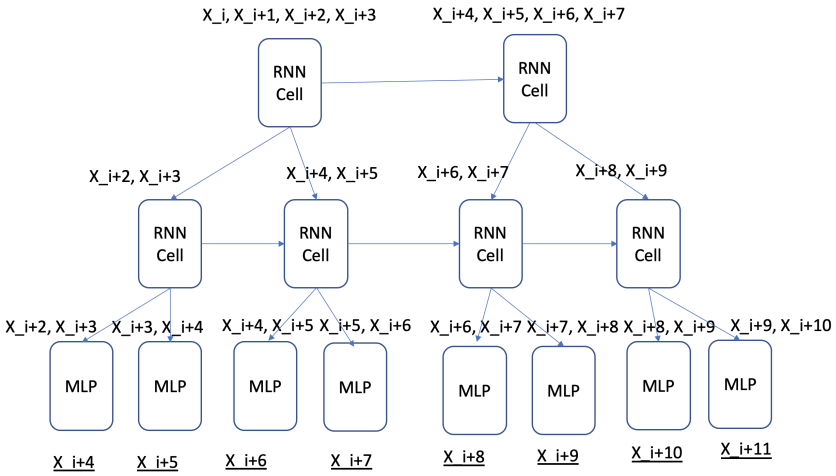

Figure 8: vanilla sampleRNN architecture

In our audio super resolution scenario, each sample (such as X_{i+4}) is generated also conditioning on its low resolution sample ($Y_{\frac{i+4}{2}}$) at the given time stamp. As shown in Fig. 9, So we feed the corresponding low resolution version of each sample into the multilayer perceptron. Note that the perceptrons of both X_{i+4} and X_{i+5} correspond to the same lower version samples $Y_{\frac{i+4}{2}}$.

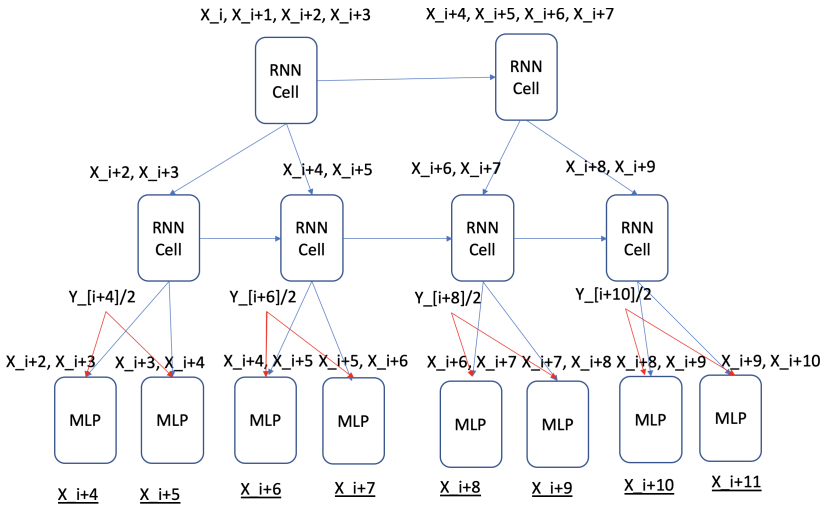

Figure 9: conditional sampleRNN architecture

8.2 Experiment design

In our proposed experiments, we train the conditional sampleRNN model (Fig.9) from scratch on VCTK data. We will implement the conditional sampleRNN based on its vanilla implementation [9]. We truncate each the high resolution .wav file (16k Hz) into non-overlapping frames, each of which consists of 1024 audio samples. Thus, each non-overlapping frame can be seen one training sample in a batch during one training step. And we keep feeding the frames to the conditional sampleRNN until they cover the entire .wav file. Meanwhile, we shuffle around .wav files to create batches. For the low resolution (4k Hz downsampled by 4 from high resolution), we first replicate each audio samples 4 times to make it align with the high resolution .wav file and also truncate them non-overlapping frames. We feed each low resolution frame with the same length into the multilayer perceptrons in conditional sampleRNN.

In inference, we feed one low-resolution .wav to conditional sampleRNN network frame (1024 audio samples) by frame. The trained network will generate each high resolution sample sequentially with in one frame and generate next frame afterwards.

For this conditional sampleRNN baseline, we also use PSNR between generated high resolution and the ground truth as the objective metric and also show their spectrums as the visualization.

References

[1] Volodymyr Kuleshov, S Zayd Enam, and Stefano Ermon. Audio super resolution using neural networks. arXiv preprint arXiv:1708.00853, 2017.

[2] Kaiming He Chao Dong, Chen Change Loy and Xiaoou Tang. Image super-resolution using deep convolutional networks. IEEE Trans. Pattern Anal. Mach. Intell., 38(2):295–307, February 2016. ISSN 0162-8828., 2015.

[3] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. Waveglow: A flow-based generative network for speech synthesis. CoRR, abs/1811.00002, 2018.

[4] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron C. Courville, and Yoshua Bengio. Samplernn: An unconditional end-to-end neural audio generation model. CoRR, abs/1612.07837, 2016.

[5] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. arXiv preprint arXiv:1609.03499, 2016.

[6] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In Advances in Neural Information Processing Systems, pages 10215–10224, 2018.

[7] Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C Cobo, Florian Stimberg, et al. Parallel wavenet: Fast high-fidelity speech synthesis. arXiv preprint arXiv:1711.10433, 2017.

[8] Resampy. Resampy upsampler https://resampy.readthedocs.io/en/0.1.5/_modules/resampy/filters.html.

[9] samplernn pytorch. https://github.com/deepsound-project/samplernn-pytorch.