CS230

Upscaling Audio Quality with Deep Convolutional Networks

Daniel Semeniuta
Department of Computer Science
Stanford University
dsemeniu@cs.stanford.edu

Abstract

The field of generative modeling and within it super resolution focuses highly on generative adversarial models. This project implements a fully convolutional network with residual connections as proposed by Kuleshov et al. for the purpose of audio super resolution. Additionally, I propose and implement a novel variation on Kuleshov et al.'s model.

1 Introduction

Advancements in deep neural networks have led to the expansion of generative models and consequently of super resolution, in which neural networks of varying architectures upscale low resolution samples of various media. Applying these techniques to the super resolution of audio is one subset of the possibilities. This project explores the problem of upscaling low quality audio samples to match the resolution of high fidelity audio data. The impact of such a project can be found in the realm of data compression, especially in the streaming era, as well as helping recover low quality audio data.

Specifically, I implement the model described by Kuleshov et al. [2017] in PyTorch (Paszke et al. [2017]), as well as propose my own novel variation on their architecture. My model takes as input a low resolution audio wav file, which is preprocessed and converted to a tensor format then upscaled via a cubic spline, and returns a high resolution version of the same audio. The reason for the spline upscaling is so that my input and output are of the same length in number of samples.

The input audio signal moves through a series of convolutional downsampling and upsampling blocks with a bottleneck architecture. In each layer, the length of each sample is halved while the its depth in terms of convolutional filters is doubled until reaching the bottleneck layer. After the bottleneck, each upsampling block doubles the length of the sample while halving the number of filters.

2 Related work

Super resolution is a very rich niche within the field of generative modeling in deep learning. Many different techniques have been applied to the problem of creating high resolution data from low resolution data. Previous generative audio models have utilized recurrent structures such as SampleRNN proposed by Mehri et al. [2016]. Their model utilizes a multi-tier, fully recurrent architecture. Work by Li et al. [2015] proposes a deep neural network based approach for bandwidth expansion in speech. Their model performs feature extraction on the input audio prior to passing the features through a DNN to predict a fuller span of frequencies from the narrow band audio.

This work instead follows the trajectory provided by research done on image super resolution. Dong et al. [2016] propose a super-resolution convolutional neural network (SRCNN) for recovering a

CS230: Deep Learning, Winter 2019, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

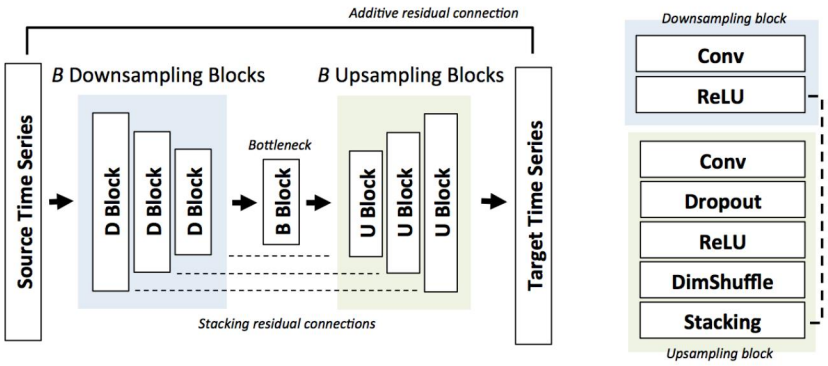


Figure 1: Architecture of Kuleshov et al.'s network

high-resolution from a single low-resolution image. Their convolutional structure was designed with simplicity at the forefront, yet outperformed many state of the art algorithms at the time.

3 Dataset and Features

The primary data utilized in this project came from the Centre for Speech Technology Voice Cloning Toolkit (VCTK) (Veaux et al. [2017]). This corpus consists of recordings of 109 English language speakers of various accents uttering around 400 sentences each for a total of 44 hours of audio data. This dataset provides a simple and consistent corpus of audio to train on. The variety of accents which is useful to speech recognition tasks provides some variety in the sound to increase difficulty and improve the ability of the model to generalize.

Using preprocessing code provided by Kuleshov et al.'s GitHub repository[1], I generated low-resolution signals of the VCTK dataset with an order 8 Chebyshev type I low-pass filter. These low-resolution signals were then upsampled via a baseline cubic spline method. All audio preprocessing was performed using packages provided with SciPy (Jones et al. [2001–]) and Librosa (McFee et al. [2019]). In the data prep stage, all VCTK data was downscaled by a factor of 4.

4 Methods

As stated earlier, I implemented both the structure suggested by Kuleshov et al., as well as my own variation on the upsampling blocks and residual connections.[2] I first describe Kuleshov et al.'s structure before discussing my variations.

4.1 Audio Super Resolution

Kuleshov et al. describe a deep convolutional network with residual connections as seen in figure 1. The network is primarily parameterized by B the number of successive downsampling and upsampling blocks. Each downsampling block contains a convolution, batch normalization, and ReLU non-linearity. Observing their source code, however, reveals that Kuleshov et al. did not use batch normalization and utilized a Leaky ReLU with negative slope of 0.2 as an activation function. In my implementation of the model, I utilized batch normalization and a leaky ReLU in my downsampling block. Each down block $b = 1, \ldots, B$ contains $\min\{2^{6+b}, 512\}$ filters of size $\max\{2^{7-b} + 1, 9\}$ with stride 2 and same padding, to achieve the desired halving and doubling of the time dimension and number of filters respectively. Due to its convolutional structure, an input of any length can be run through the model.

After the bottleneck convolution, the upsampling blocks perform the inverse of the downsampling blocks. The number of filters and the size of each one that the bth upsampling block has corresponds to the number of filters and their size that the $B - b + 1$th downsampling block had, so that they may

[1] https://github.com/kuleshov/audio-super-res

[2] Source code can be found at https://github.com/d-semeniuta/cs230-final

match in dimensionality for the residual connection. Again same padding is used, but the stride is of length 1 so the convolution does not reduce the time dimension. Again, a ReLU non-linearity is applied along with a dropout layer after the convolution.

The input passed in to a given upsampling block is of size $F \times d$ where F is the number of channels the signal has and d is the length of the audio input in its time dimension. At the end of the convolution, the input is of size $F/2 \times d$. However, the residual from the corresponding downsampling block is of size $F/2 \times 2d$. As we wish to concatenate the filters of these samples along the time dimension, we must manipulate the dimensionality of the convolution output.

Kuleshov et al. suggest a one-dimensional version of a Subpixel layer proposed by Shi et al. [2016]. The original two-dimensional subpixel layer shuffles an input of size $F \times d \times d$ to one of size $F/4 \times 2d \times 2d$. As PyTorch only provides a two-dimensional implementation, I implemented my own one-dimension subpixel shuffle which transforms our convolution output from size $F/2 \times d$ to $F/4 \times 2d$. This output is then concatenating with $F/4$ features from the downsampling residual to form a final output of the upsampling block of size $F/2 \times 2d$. Though Kuleshov et al. provide no direct hints for deciding which $F/4$ features to choose from the downsampling residual, I implement this with a linear projection. We have thus halved the number of filters and doubled the time dimension.

A final residual connection occurs after a final convolution following the upsampling blocks. This final convolution results in a signal of the desired length and a single channel. So that the model does not need to learn the full reconstruction of the original input signal, but simply the difference between the low-res input and high-res prediction, there is an additive residual connection between the input signal and the final layer.

Evaluating the quality of an audio approximation as compared to a reference high-resolution audio sample leads to a straightforward use of a mean squared error (MSE) objective function

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

for optimizing the parameters θ of our network f, where x_i, y_i are the pairs of low-resolution and high-resolution audio in our dataset.

4.2 Audio Super Resolution V2

I propose a novel iteration on Kuleshov et al.'s model. The driving motivation is to utilize the same logic of learning the difference between the source and the target and creating an additive rather than a stacked residual connection between the downsampling and upsampling blocks. To accomplish this, I double the number of filters in the convolutional layer of the upsampling blocks, so the input after convolution is of size $F \times d$. Performing the one-dimensional subpixel shuffle transforms it to a signal of size $F/2 \times 2d$, matching exactly the shape of the residual. In this case then, there is no need to project the residual or choose which $F/4$ channels of it to use.

5 Experiments/Results/Discussion

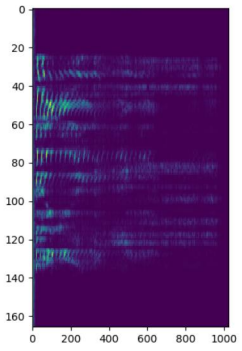
The model Kuleshov et al. presented utilized the number of filters and filter size stated above in section 4.1. Additionally, their tested model had $B = 4$ blocks. In addition to training a model utilizing their hyperparameters, I trained larger models to improve the robustness and quality of the model. I increased the number of blocks to $B = 5$ as well as increasing the maximum number of filters for a layer to 1024. For my suggestive additive residual model, I additionally lowered the number the initial number of filters so my model had $\min\{2^{5+b}, 1024\}$ filters of size $\max\{2^{7-b} + 1, 9\}$. I made this change to decrease the jump their model had from a single channel audio file at input to 128 channels in the second layer, as there may not be enough information in the signal this early in the network.

The model parameters were optimized as stated earlier with an MSE objective by an Adam optimizer with learning rate of 5×10^{-5} and $\beta_1 = 0.9, \beta_2 = 0.999$ in all variations of hyperparameters. The models were trained for between 50-100 epochs depending on the hyperparameter initialization.

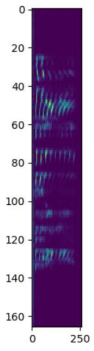
Although MSE provides a good objective to parameterize against, it is not indicative of the success of the model. For evaluating the quality of the audio reconstruction the Signal to Noise Ratio (SNR)

Residual	Max Filter Size	B	Epochs Trained	SNR	LSD	MSE
Stacked	524	4	50	45.8967	1.19825	4.1263×10^{-5}
Stacked	524	5	50	45.9406	1.18799	3.78957×10^{-5}
Stacked	1024	5	110	48.5504	1.09285	3.27255×10^{-5}
Additive	1024	5	80	45.8967	1.19825	4.1263×10^{-5}

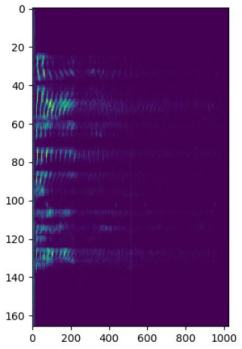
Table 1: Model Results on Validation set



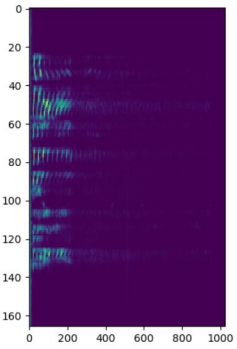
(a) Reference Spectrogram



(b) Low Resolution Downsample



(c) Paper Parameters Prediction



(d) Additive Residual Prediction

Figure 2: Spectrograms of speaker 225 utterance 366 in VCTK corpus

and Log-spectral distance (LSD) (Gray Jr and Markel [1976]) were utilized. The SNR is a standard metric for measuring the level of a desired signal, in this case audio, to background noise. Given a reference signal y and an approximation y it is defined

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

LSD measures the reconstruction quality of individual frequencies with the formulation

$$\text{LSD}(x, y) = \frac{1}{L} \sum_{\ell=1}^{L} \sqrt{\frac{1}{K} \sum_{k=1}^{K} \left(X(\ell, k) - \hat{X}(\ell, k) \right)^2}, \tag{3}$$

where X and $\hat{X}$ are the log-spectral power magnitudes of y and x, respectively. These magnitudes are defined as $X = \log |S|^2$ where S is the short-time Fourier transform (STFT) of the signal x or y. This is parameterized by the number of index frames and frequencies in the experiments, denoted by ℓ and k, respectively.

Table 1 displays the results achieved by the various models trained. Though the metrics look very good, I believe there is a bug in my calculation of both SNR and LSD. The numbers achieved are very out of the ordinary, but the models do not hold up to subjective listening. All predictions made

by the models, including the one utilizing Kuleshov et al.'s hyperparameters, sounded very poor in comparison to the high resolution reference signal. Due to the very low loss and good metrics reported during training, along with a seeming plateau, I stopped training my models far earlier than the 400 epochs. Figure 2 displays spectrograms of the audio samples, where the y-axis representes the frame of the signal, and the x-axis represents the frequency. It is apparent that the models do not accurately reconstruct the higher frequencies of the reference audio signal given the low resolution downsample.

6 Conclusion/Future Work

The model unfortunately did not succeed in reconstructing high quality audio from low-resolution inputs. I believe bugs in the implementation of the metrics led to misleading results which further led to stopping the models from training earlier than they should have been stopped. The model shows extreme promise in its structure as the convolutional architecture allows for inputs of any size. In these early results, the bigger models which trained for longer showed the most promise.

These results indicate that future work should focus on resolving the issues regarding the metrics and simply training the model for longer. Beyond that, I would experiment with the architecture more, utilizing different numbers of filters and different sized filters in each convolution. I would also iterate on various hyperparameters of Leaky ReLU or experiment with a simple ReLU as an activation function after the convolutions. Additionally, in the stacked residual model, I believe there is a better way to reduce the $F/2$ channels to $F/4$ instead of a fully connected linear layer, as this may reduce the positive effect of a residual. One-dimensional point-wise convolutions may fit better within the fully convolutional model scheme, or simply utilizing max pooling or average pooling with a pooling window of 2 would reduce the number of channels.

Additionally, it would be interesting to incorporate this model with other aspects of generative modeling, such as an adversarial network.

This project is simply the start of exploration within generative audio for super resolution. There is more work which can be done to fully realize the potential of this model and within the field.

Contributions

Alejandro Ballesteros assisted in the initial proposal and milestone stages of the project. However, he withdrew from the course, and the code and final project write up were entirely done by Daniel Semeniuta.

References

C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, Feb 2016. ISSN 0162-8828. doi: 10.1109/TPAMI.2015.2439281.

A H. Gray Jr and J D. Markel. Distance measures for speech processing. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 24:380 – 391, 11 1976. doi: 10.1109/TASSP.1976.1162849.

Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL `http://www.scipy.org/`.

Volodymyr Kuleshov, S. Zayd Enam, and Stefano Ermon. Audio super resolution using neural networks. *CoRR*, abs/1708.00853, 2017. URL `http://arxiv.org/abs/1708.00853`.

Kehuang Li, Zhen Huang, Yong Xu, and Chin-Hui 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.

Brian McFee, Matt McVicar, Stefan Balke, Vincent Lostanlen, Carl Thomé, Colin Raffel, Dana Lee, Kyungyun Lee, Oriol Nieto, Frank Zalkow, Dan Ellis, Eric Battenberg, Ryuichi Yamamoto, Josh Moore, Ziyao Wei, Rachel Bittner, Keunwoo Choi, nullmightybofo, Pius Friesch, Fabian-Robert Stöter, Thassilo, Matt Vollrath, Siddhartha Kumar Golu, nehz, Simon Waloschek, Seth, Rimvydas

Naktinis, Douglas Repetto, Curtis "Fjord" Hawthorne, and CJ Carr. librosa/librosa: 0.6.3, February 2019. URL https://doi.org/10.5281/zenodo.2564164.

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. URL http://arxiv.org/abs/1612.07837.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1874–1883, June 2016. doi: 10.1109/CVPR.2016.207.

Christophe Veaux, Junichi Yamagishi, and Kirsten MacDonald. Cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit, 2017. URL http://homepages.inf.ed.ac.uk/jyamagis/page3/page58/page58.html.