Audio Noise Elimination Using A Transformer Based Recurrent Neural Network

Landon Burleson
Electrical and Computer Engineering
Oklahoma State University
Stillwater, Ok, USA
landon.burleson@okstate.edu

Madhusti Dhasaradhan
Electrical and Computer Engineering
Oklahoma State University
Stillwater, Ok, USA
madhusti.dhasaradhan@okstate.edu

Alex Sensintaffar
Electrical and Computer Engineering.
Oklahoma State University
Stillwater, Ok, USA
alex.sensintaffar@okstate.ed

1. Abstract

Denoising an audio track without altering the integrity of voice data is a complicated task within the field of digital signal processing. Common implementations within industry include digital filtering using techniques such as wiener filtering or other non neural network strategies. Some neural network implementations include LSTM (Long Short Term Memory) or the simplified equivalents, GRU (Gated Recurrent Unit). However, these implementations, while do help compute relationships between the time samples, do so with limited success with long time-series sequences. To avoid the limitations of LSTM-based neural network implementations, we are proposing a transformer based RNN model to be used in a random and environmental audio denoising model. The transformer based model also has the added characteristic of the ability to be parallelized to maximize throughput and speed to the final output of the neural network model. This speed would allow for real-time audio to be inputted into the trained model and output filtered data with a minimal delay between the input and output. Utilizing both the transformer and RNN based architectures, we will be able to show the empirical improvements when utilizing the transformer in comparison to the RNN architecture.

2. Introduction

Noise in audio is detrimental to a listener in terms of understanding and enjoyment. Filtering the undesirable frequency ranges in real-time has proven to be computationally heavy in terms of processing power required to achieve this goal. Graphics card manufacturer, Nvidia, has implemented such a feature for their former generation of Graphics Processing Units (GPU) and beyond by leveraging the bandwidth available in the aforementioned processors. However, using the real-time filtering available on this hardware requires immense power and proprietary algorithms. The goal for this project is to propose an open-source Neural Network architecture that is efficient enough to satisfy the real-time filtering offered elsewhere on various hardware. This problem is interesting due to the complexity of accounting for speed of the neural network and the approach in which to tackle voice recognition within audio tracks or audio channel(s).

Utilizing an RNN and transformer based architectures, we will be able to filter environmental audio and background noise in real-time for better clarity to a listener in a virtual meeting / conference setting. The primary datasets that will be used to accomplish this will be the MS-SNSD[1] and our own random noise induced audio tracks with single words being spoken. Our own constructed dataset will be discussed further in later sections of this report. Existing implementations utilized LSTM[2] based RNNs. We propose utilizing a transformer based RNN structure to modernize the architecture and allow our model to more easily adapt to long time-series sequences without hindering performance.

3. Background

A recent paper utilized a LSTM based RNN architecture to provide recognition of voices and similarity between samples[2] . Other implementations, such as Nvidia's, are proprietary and thus encompassed in an unreachable 'black box'. Utilizing these references, we will innovate with an open-source, low overhead transformer based RNN model.

Other aforementioned work in this digital signal processing area includes traditional non-ML based solutions. Among these techniques include the wiener filtering methodology. However, this solution has the drawback of over filtering the signal and thus degrades the integrity of the desired audio data.

4. Approach

As mentioned before, our proposal to tackle the stated problem, is to implement a transformer based RNN in order to better avoid the vanishing gradient problem as well as take advantage of the throughput benefits the Attention module has to offer within this type of architecture. These throughput benefits would allow us to input a live audio stream into the model and process the filtering of both

[1] https://github.com/microsoft/MS-SNSD

[2] Z. Li and P. Song, "Audio similarity detection algorithm based on Siamese LSTM network," 2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP), 2021, pp. 182-186, doi: 10.1109/ICSP51882.2021.9408942.

background noise and environmental noise from the audio stream for a better experience on the listeners side. This technology is both useful and highly desired in conferences, phone calls, and pertinent communications that have extreme background audio levels such as in industrial environments.

The attention module within the transformer would focus upon our desired data and thus reducing the overall interference noise would have on the input audio source.

The primary data that will be used will stem from the MS-SNSD dataset. This dataset has both clean and noisy voice audio files that can be used to train our proposed neural network architecture to filter noise and focus on speech patterns. Other datasets that may be used is the audio MNIST dataset that has short audio recordings of low-noise voices reading simple sentences. This could be used in conjunction with the MS-SNSD dataset to better recognize voices within our network architecture. See the footnotes for hyperlinks to both of the described datasets.

The data was later changed to the tensorflow mini_speech_commands data set. This data is small 1 second sounds and easy to use. The data consist of 8 words down, up, go, stop, left, right, no and yes. Each word is spoken 1000 time by different voices and each stored as a separate .wav files. For each .wav file 10 noisy copies are created. The resulting 80,000 files are about 1.6 GB in total data. To reduce the overall data burden we will only use 100 MB of data. This will result in approximately 5,000 files being used. 3,000 files will be used to train, 1,000 files will be used to test, and 1,000 files will be used for verification.

The random noise was created based on a normal Gaussian distribution. The root mean square of the signal is used to determine the scale of the distribution and the shape of the signal is used to determine the size of the distribution. When played the voice saying the word can still be heard but there is clear and noticeable static in the background.

$$RMS = \sqrt{mean(signal^2)}$$

Equation 1. RMS Equation used to find the RMS of the sound file

The following graphs are used to showcase a signal with and without noise for the same audio track. Notice the noise encompasses the lower amplitude details and leaves the relatively high amplitude portion of the audio relatively untouched. By observation, the original signal is difficult to discern without viewing the original audio track spectrum. In this use case, it would be difficult to utilize standard DSP filtering techniques as the noise is random and populates the entire frequency spectrum. The attention module inside our proposed model will yield excellent results recognizing voice data versus the noise as those features will be the most prevalent to the model when compared against the clean audio versus the noisy audio. The process in which we will be evaluating performance will be discussed in the next section of the report.

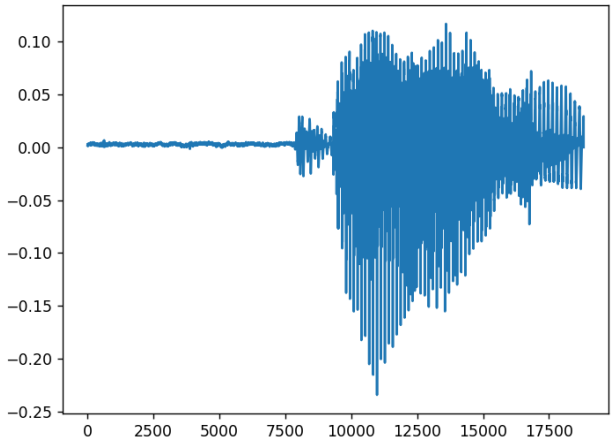

Figure 1. Initial Sound Without Noise

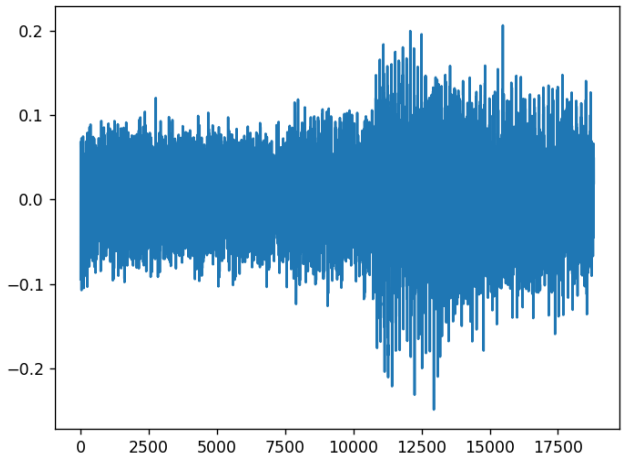

Figure 2. Sound File with Random Noise Added

Sound File Without Noise
Sound File With Noise

4.1 Evaluation

Our proposed model will be evaluated using the audible quality. This would be done by directly comparing the input and output manually. The scipy module in python also provides audio-based analysis to examine the frequency domain of the audio tracks. Another metric to measure performance is the delay needed to make a computation between a transmitted stream (speaker) and the received stream of audio (listener). Another quantitative measure is the signal-to-noise ratio that can be measured using the scipy python module.

In addition to the manual audio input vs output review, among the metrics that will be used to distinguish the difference between the clean audio sample and the environmental/ random noise input sample will be the mean squared error (MSE). This metric is often used to find the similarities between two sets of regressive data. With the

datasets being a regressive dataset rather than a categorical type, this metric will easily find the closeness between the actual and the predicted sequential data samples.

5. Preliminary Results

The following graph showcases the initial MSE versus epoch graph for a single audio track in the learning process.

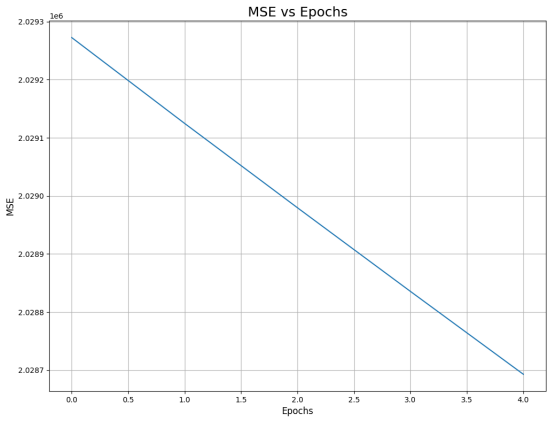

Figure 3. Single Track Training MSE vs Epochs

The above result represents the MSE values from inputting an audio track with approximately 18k data points within the track and over the course of 5 epochs respectively. Note the linearity of the graph. This outcome is due to only using a single audio track rather than multiple tracks to properly train the network. At the time of writing, the desired datasets were not in a format to provide a complete preliminary test of the baseline RNN. See Figure 4 to view the initial RNN structure for use in our baseline comparison between the RNN and transformer architectures.

Due to utilizing the MSE for the loss optimizer, the Loss vs Epochs graph from the same training run as shown above in Figure 3, the graph for Loss vs Epochs materializes in the same linear characteristic. Furthermore, this graph has been omitted from the report until more rigorous training for the proposed model and the baseline is completed.

As shown in Figure 4, the initial structure of the baseline RNN is shown with two SimpleRNN and Dense layers each. This structure, as well as the number of neurons/cells in each, are subject to change in accordance with the training and debugging portion of this project. The hyperparameters used will also be adjusted to maximize the denoising within the input signal while still maintaining the voice integrity in the audio track. This also includes the delay in which the input data is converted from noise to a clean audio final output. This delay will ultimately be the driving force for the feasibility of a real-time audio filtering model and is thus crucial to adjust the model parameters with this constraint in mind.

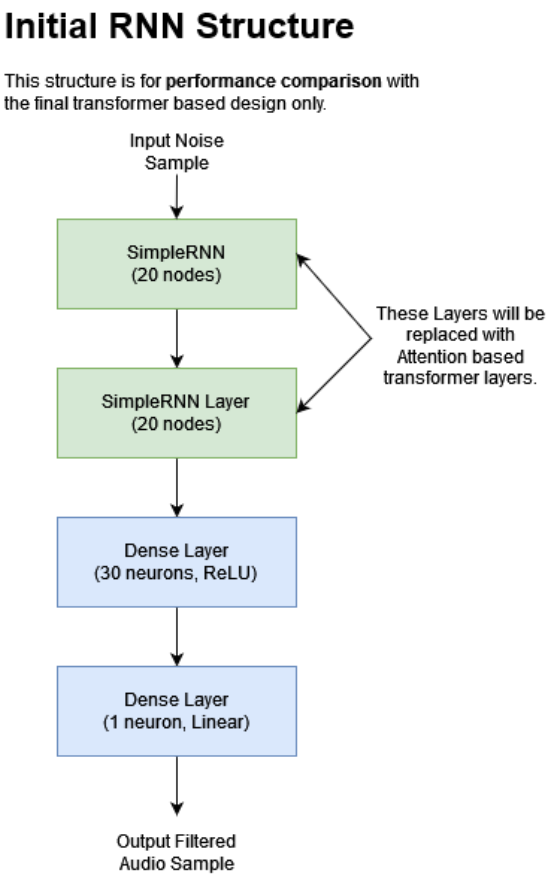

Figure 4. Baseline RNN with Annotations needed to Alter the Architecture to a Transformer.

The above structure is used to provide a baseline for our project by utilizing an existing architecture used in DSP applications for comparison against the state-of-the-art transformer based design that we are proposing. The transformer based design is currently being tested and debugged in order to meet future deadlines. All hyperparameters shown in the report are subject to change as we as a team find better performing models than currently shown.

Given the amount of time left for the project prior to submitting the deliverables, the prospect of a excellently performing real-time audio filtering model seems to be unrealistic. To adjust the scope of our project, we propose a audio filtering transformer based model for archival or premade audio tracks rather than live data. This would allow us to focus more on the quality of the filtered tracks and adjusting the hyperparameters of the model as such. However, if the measured delay between the input samples and the final converted output from the model is low enough, we will also include this test case in the final report as well as a live recorded sample for a real-time audio test.

The following section describes our past and future timelines broken down for each week and team member.

5.1 Timeline

The following table represents our tentative timeline.

Time	Landon	Madhusti	Alex
Week 1 (10/28)	Prior Work Investigation	Prior Work Investigation	Prior Work Investigation
Week 2 (11/4)	Data collection / cleanup	Data collection / cleanup	Data collection / cleanup
Week 3 (11/11)	Architecture Design	Architecture Design	Architecture Design
Week 4 (11/18)	Test / Debug	Test / Debug	Test / Debug
Week 5 (11/25)	Test / Debug	Test / Debug	Test / Debug

To date, we as a group have collected all data samples we will use to train and test our model. We have created many data points to split the testing and training data with the relative number of points being referenced in the approach section above. As of now, we have developed a simple RNN design to use for final comparison purposes with the final transformer based design. As of writing, the training and testing of the model will become the main priority for the next two weeks of the schedule as shown in the above timeline table. Tentatively, we will be working with both the simple RNN layers and work with the transformer layers in parallel in order to meet the final deadlines and criteria.

6. References

[1] C. K. A. Reddy, E. Beyrami, J. Pool, R. Cutler, S. Srinivasan, και J. Gehrke, 'A Scalable Noisy Speech Dataset and Online Subjective Test Framework', *Proc. Interspeech 2019*, σσ. 1816–1820, 2019.

[2] Z. Li και P. Song, 'Audio similarity detection algorithm based on Siamese LSTM network', στο *2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP)*, 2021, σσ. 182–186.

[3] Mart'in Abadi *κ.ά.*, 'TensorFlow Large-Scale Machine Learning on Heterogeneous Systems'. 2015.

[4] Mark Daoust *κ.ά.*, 'TensorFlow Simple audio recognition Recognizing keywords'. .