CS230

Audio Separation and Isolation: A Deep Neural Network Approach

Ahmed Hamdy ahamdy@stanford.edu
Pratap Kiran Vedula pvedula@stanford.edu
Muni Venkata Jasantha Konduru jkonduru@stanford.edu

Abstract

A particular sound of interest is almost always overlapping with other multiple waves from different sources, and if those are at a comparable if not higher amplitude, will lower the ability to interpret effectively when heard. Audio separation and isolation will allow us to focus on specific sounds of interest. The motivation of the project is to able to separate and isolate a mixed audio file using deep learning. The CNN autoencoder model implemented consists of an encoding stage with two convolution layer followed by a single fully connected layer followed by a decoding stage consisting of an array of fully connected layers and two deconvolution layers per class.

1 Introduction

Sound event separation and isolation requires a trained system, when presented with an unknown sound, to correctly identify and isolate it. We propose a solution to what is commonly referred to as the "cocktail party problem". The model implemented would learn and apply different filters to an input in order to obtain a set of audio sources from a mixed audio input. Few example scenarios of what we have attempted to enable are self-driving cars identifying a police siren, isolating a broadcaster's voice from others in a loud crowd, or recognizing an infant crying in a noisy environment. The challenging aspect of this project was to train with a larger dataset of audio files. We opted for pre-processing of entire dataset at once and ran in to memory limitations, then we opted for batch processing. The input of our algorithm is a mixed audio file which consists of different sounds overlaid. The output is classified to five individual isolated audio files. We relied on PyTorch framework to build this mode and used STFT as a feature extraction and min-max for data normalization.

2 Related work

Our model attempts to solve the source separation problem by basing its approach to one similar to that discussed by Chandna, Pritish et al. [7] with their musical separation model implementation, though was built from scratch due to the scarcity of implementations that tackle the same cocktail party problem as well as ease of code interpretation and modification. Research done by Stefan Uhlich, Antoine Liutkus and Yuki Mitsufuji [10] proposes implementing Open-Unmix for music source separation and the task to decompose music into its constructive components such as stems of vocals, bass, and drums. This was considered to be the state-of-the-art model [11] to be referred to for implementation comparisons.

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

3 Dataset and Features

We researched on various options for data generation that will allow us to create input and output audio files for audio separation and isolation. Main audio categories used are Music, Human, Human Speech, Nature, Effects, Urban, and Noise that are available from Making Sense of Sounds Challenge 2018 database [1], the Spoken Wikipedia Corpora (SWC) [2]. Siren, Baby Cry, and Dog audio files were obtained from Kaggle [5] and GitHub [2][3].

Audio files for the training and dev sets were synthesised by taking 4 individual sounds corresponding to the classes (1) Baby Cry, (2) Siren, (3) Dog, and (4) Human Speech from above mentioned audio categories and overlay them together to create the input sound. We then overlaid 2 randomly chosen additional audio files to the input audio file for class (5) Other. All individual audio files are correspondingly classified as output classes. As part of the dataset generation, each audio file is processed into 5 sec chunks, pitch randomly shifted by 2 or 4 steps, randomly stretched by 1.2 times, and loudness shifted by a randomly chosen value between -5 and +5 points. This ensures that the model will be trained and evaluated on a robust data set. Generated audio files are 5 seconds long and encoded using PCM codec in WAV format, with mono channel and sampled at 44100 Hz. Table 1 below show initial individual audio file classes used to generate 100000 file dataset and the percentage duplicate counts in generated dataset. We notice that dataset generated is diverse, with minimal duplicates and overlap.

Number of Audio files per Class that were used for Dataset generation					
	Crying Baby	Siren	Dog	Speaker	Other
Reduced MSoS + SWC	31	16	16	2540	300
MSoS + SWC + Baby + Siren + Dog + Augmented (Baby, Siren, Dog)	111	336	440	2540	1442

Percentage of Duplicate across 100000 Dataset per Class					
	Crying Baby	Siren	Dog	Speaker	Other
Reduced MSoS + SWC	1.935483871	3.75	3.75	0.023622047	0.2
MSoS + SWC + Baby + Siren + Dog + Augmented (Baby, Siren, Dog)	0.540540541	0.178571429	0.136363636	0.023622047	0.041608877

Table 1: Tabular representations of the number of audio files per class before Dataset generation and the percentage of duplicates across 100000 Dataset

Prepossessing is performed by converting input and output audio files to 22050 Hz, reducing the duration of audio files to 3 seconds, normalize using MIN MAX parameters, and extracting STFT from the input and output files. When transforming the audio, a STFT window size of 23 milliseconds and a hop length that's fourth the size of window is used. Figures 1 and 2 below demonstrate the STFT spectrogram representation of a sample input and output files. We used 100000 files as part of our dataset. We leveraged 92 percent of the data for train set and remainder 8 percent as dev set.

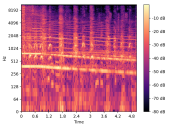

Figure 1: STFT spectrogram of sample input audio file

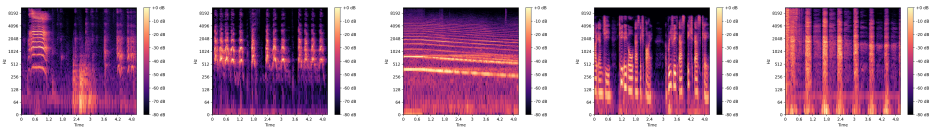

Figure 2: STFT spectrogram of sample output audio files representing Baby Cry, Siren, Dog, Human Speech, and Other respectively

4 Methods

We implemented an encoding stage which consists of two convolution layers followed by a single fully connected layer and a decoding stage replicated in parallel for each of our five classes which

consists of a single fully connected layer followed by two deconvolution layers as presented in Figure 3 below.

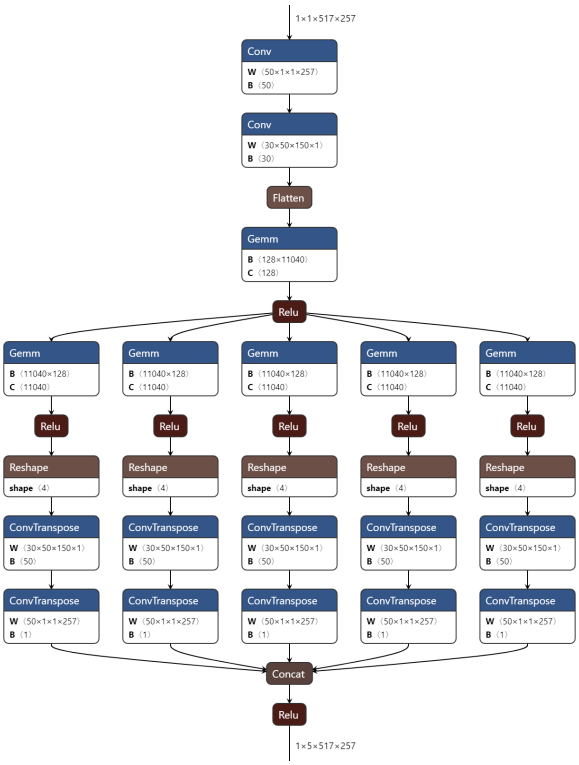

Figure 3: Illustrative representation of the model architecture

The layers in the encoding stage correspond to (1) a vertical convolution layer in order to obtain the frequency feature, (2) a horizontal convolution in order to obtain the time dependent feature and thus outputting a time-frequency encoding, and (3) a fully connected layer with a Rectified Linear Unit (ReLU) as the activation which shares its output with the layers in the decoding stage. The layers in each of the decoding stages per four classes correspond to (1) a fully connected layer that shares the same dimensions as the encoding horizontal convolution layer for minimal information loss with a ReLU as its activation function, as well as (2) a horizontal and (3) vertical deconvolution layers which share the opposite dimensions of the horizontal and vertical convolutions layers in the encoding stage. The output of each of the five decoding stages is then concatenated for a ReLU to be applied to before being used for loss calculation and back propagation.

The model is provided with a mixed source input and five isolated source outputs, where all have gone through feature extraction and normalization, and attempts to train on the magnitude of the input STFT while disregarding its phase. It was found that instead of training on both magnitude and phase of the input STFT, training only on magnitude and passing the phase of the input to each of the five outputs provided a negligible difference to the signal quality in terms of source-to-interference ratio between (a) output signal with both its respective magnitude and phase and (b) output signal with its respective magnitude but input phase. Doing so also provided a decreased the number of parameters to train over by a factor of 2 as compared to training on both the magnitude and phase of the input STFT. The model then attempts to learn a set of five different filters that are applied to the input to obtain five outputs representing the isolated sources, which are calculated as:

- Extract each of the five classes from the concatenated model output, o_i
- Divide each of the extracted outputs by the total sum to obtain the filters $f_i = \frac{o_i}{\sum_{n=1}^{5} o_n}$
- Multiply each of the filters to the input to obtain the predicted isolated output $\widetilde{y_i} = f_i * x$

Once the filters are applied to the input to obtain the five calculated outputs, the losses are computed per class by measuring the Mean Squared Error (MSE) between the calculated and the provided

target output. The individual losses are then summed up together for the model to attempt to optimize using the ADAM [8] optimization algorithm. The Adaptive Learning Rate Method (ADADELTA) optimization algorithm proposed by Zeiler, Matthew D [12] was also experimented with and attempts to adapt the learning rate dynamically over time per parameter.

5 Experiments/Results/Discussion

During the experimentation iterative process to identify the optimal model variables, we came up with a set of parameters with which we evaluated in combination of one another for the model training. A subset of those parameters are summarized in Table 2 below, with which they contributed to the (1) size of the hidden units (FC Features, Horizontal and Vertical Conv. Filter Sizes), (2) size of the input (STFT Size, STFT Hop Length, Sample Rate), (4) loss optimization (ADAM and ADADELTA as well as their respective parameters), (4) training iterations (Epochs), (5) normalization techniques (MIN MAX and CMVN), and (5) dataset size.

Parameters	**Options**		
Optimizer	ADAM	ADADELTA	
ADAM Learning Rate	0.01	0.005	0.001
ADADELTA rho	0.98	0.95	0.9
STFT Size	1024	512	
STFT Hop Length	256	128	
Horizontal Conv. Filter Size	(1, 513)	(1, 257)	
Vertical Conv. Filter Size	(15, 1)	(350, 1)	(150, 1)
FC Features	128	376	
Normalization	MIN MAX	CMVN	

Table 2: Tabular representation of the subset of different parameters iterated over

Several combinations of those parameters were then investigated with which a subset provided desirable results. The bayes optimal error in this case is one where the avg. loss is 0 and with further evaluation we determined the human level to be an avg. loss of around 35 for an isolated output with an acceptable interference and signal quality reduction. The base model with which we started with was built with zero padding and convolution layers as well as optimized the loss on the model output directly. Using this model produced a train avg. loss output of 250000 which is significant and entails a massive bias issue. The purpose of the zero padding along with the convolution layers in the decoding stage was to maintain the size of the waveform input in order to recreate the isolated output. Maintaining the same purpose, we realized that using deconvolution layers is a more preferable method to approach this as it attempts to reverse the convolutions in the encoding stage. This decreased our train. avg. loss to 40000, which was a desirable drop, though still far from the ideal. We then attempted to optimize the loss of the model on the filter of its output as proposed by Chandna, Pritish et al. [7] and described in the earlier section and we were able to observed a train. avg. loss of 1700. We then applied ReLU to the concatenated model output and used ADAM [8] as our loss optimizer and we started seeing the train avg. loss decrease to 240 while observing a dev. avg. loss of 272. With this we were able to chip into our bias issue though now observe a variance issue. Focusing on decreasing the bias for the time being, we attempted to optimize training by using CMVN instead of MIN MAX as a normalization technique, which resulted in a train avg. loss of 219 and a dev. avg. loss of 204 and thus decreased our bias further though now realize that our dev. set might be easier to solve with its decreased loss compared to the train avg. loss. Observing the dataset distribution of classes, we identified that approximately 5% the waveforms mixed for each of classes (except for Other) appear to be populated with useful data while the remaining 95% is silence. This can be attributed to how on random our dev set seems to be easier to predict especially if a large percentage of it is based on silence, considering that we were training on a large number of silent waveforms. Mediating this issue was possible by making sure that there is a 70/30 split of waveform data to silence for each of the classes (except for Other).

Following that with several sets of experiments, the final model which was settled upon was then equivalent to the one detailed in the earlier section, which in turn provided us with a train avg. loss of 60 and a dev. avg. loss of 129, implying that the model is overfitting as we see a variance issue and necessitating a bigger dataset to train on to remedy this issue. Table 3 below presents a summary of the subset of experiments evaluated.

For the model prediction evaluation, we are using the mir_eval metric as proposed by Raffel, Colin, et al. [9] as the library provides us with the capability of evaluating an estimated source to a target one

	Experiments	Train Avg. Loss	Dev. Avg. Loss
A	Exchanged Zero Padding + Convolution Layers with Deconvolution Layers	40000	-
B	Training on Filters of the Model Output	1700	-
C	Using ADADELTA as Loss Optimizer and apply ReLU to the concatenated model output	240	272
D	Exchanged MIN MAX with CMVN Normalization	219	204
E	Increased size of FC Features to 376	237	225
F	Exchanged CMVN with MIN MAX and ADADELTA with ADAM at LR 0.001 as well as decreased size of FC Features to 128	66	129

Table 3: Tabular representation of the different parameters iterated over

when dealing with source separation tasks, based on the Signal to Distortion Ratios (SDR), Source to Interference Ratios (SIR), and Sources to Artifacts Ratios (SAR). Running a few of the models experimented with earlier across their respective dev sets, we can then observe the evaluation metrics per class as shown in Figure 4 below.

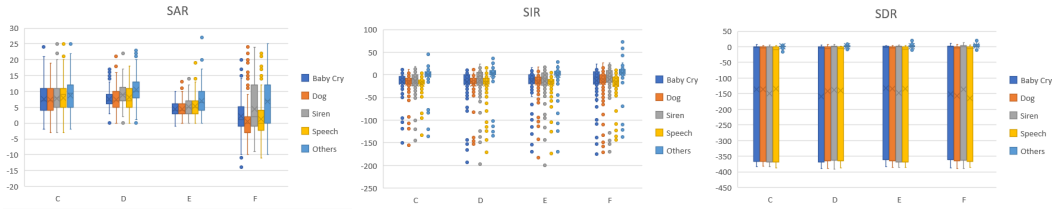

Figure 4: Evaluation metrics for each class per experimented model

We can observe that the dev. avg. losses determined earlier seem to correlate with the evaluation metrics presented here, as we can see that each of the SDR, SIR, and SAR for Model D seems to be the best compared to Models C and E. Saying this however, we can perceive that the SDR for all of the models experimented present an average of around -150, which entailing that the ground truth target is not too similar to the estimated output. The SIR presents a better results though on average appears to be around -10, entailing that the estimated output shares some portions of the waveform with the other estimated outputs, as in we will be hearing some interference amongst the classes. This however might not a big of an issue since the SAR provides us with a result of around 8 on average, meaning that the isolated output is glitch free with minimal artifacts present. Comparing our results to published implementations was difficult due to the scarcity of audio separation implementations that consider universal sounds as opposed to focusing on music isolation. Nevertheless, since the end goal is the same and the class human speech exists as vocals in those implementations, we present in Table 4 below a comparison of the results of our implementation to that of Open-Unmix [10], which was reported as the state-of-the-art model by the Signal Separation Evaluation Campaign 2018 [11]. We can see that even though our SAR is superior to the model, both the SIR and SDR deteriorate the signal isolation quality as discussed above and as compared to the Open-Unmix model.

Model	SDR	SIR	SAR
Open-Unmix	6.32	13.33	6.52
CNN Model	-139	-18.7	8

Table 4: Comparison of state-of-the-art Open-Unmix model and our implementation

6 Conclusion/Future Work

Our main objective of this project is to try to classify and isolate a mixed audio input file using STFT for feature extraction and MIN MAX for normalization. So far, we were able to achieve the expected results with five different categories, further insights are required to train the model with a larger dataset. Future work includes: (1) obtain a Mel Spectrogram representation of the input transforming the sounds onto the Mel Scale, (2) training the model on a larger data set to remedy the variance issue observed, (3) explore the use of LSTM in order to train the network on shorter waveforms for near real-time inference, and (4) accounting for discriminatory loss terms per class to further decrease SDR and SIR.

7 Contributions

Through a set of different tasks, efforts and contributions to the project have been distributed equally amongst all members of this group. Table 5 below details the tasks taken up by each team member.

Member	Task			
Ahmed	Model Research	Feature Extraction Implementation	Model Implementation	Training and Evaluation
Pratap	Model Research	Dataset Research	Dataset Generation	Training and Evaluation
Muni	Model Research	Feature Extraction Research	Normalization Implementation	Training and Evaluation

Table 5: Task breakdown per team member

References

GitHub Repository: https://github.com/ahpvjk/audio-classification-and-isolation

[1] Harris, Lara; Bones, Oliver Charles (2018): Making Sense Of Sounds: Data for the machine learning challenge 2018. figshare. Dataset. "https://doi.org/10.17866/rd.salford.6901475.v4"

[2] Köhn, A., Stegen, F., & Baumann, T. (2016). Mining the spoken wikipedia for speech data and beyond.

[3] Siren-Identification-Localization, (2016), GitHub repository, https://github.com/Siren-Identification-Localization/Siren-Identification-Localization/tree/master/datasets

[4] donateacry-corpus, (2015), GitHub repository, https://github.com/gveres/donateacry-corpus

[5] Moreaux, M. (2017, October). Audio Cats and Dogs, Version 5. https://www.kaggle.com/mmoreaux/audio-cats-and-dogs

[6] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., Devito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in PyTorch.

[7] Chandna, P., Miron, M., Janer, J., & Gómez, E. (2017). Monoaural Audio Source Separation Using Deep Convolutional Neural Networks. *LVA/ICA*.

[8] Kingma, D. P., & Ba, J. L. (2014). Adam: A method for stochastic optimization. arXiv 2014. *arXiv preprint arXiv:1412.6980*.

[9] Raffel, C., McFee, B., Humphrey, E. J., Salamon, J., Nieto, O., Liang, D., ... & Raffel, C. C. (2014). mir_eval: A transparent implementation of common MIR metrics. In In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR.

[10] Stöter, F. R., Uhlich, S., Liutkus, A., & Mitsufuji, Y. (2019). Open-unmix-a reference implementation for music source separation.

[11] Stöter, F. R., Liutkus, A., & Ito, N. (2018, July). The 2018 signal separation evaluation campaign. In International Conference on Latent Variable Analysis and Signal Separation (pp. 293-305). Springer, Cham.

[12] Zeiler, M. D. (2012). ADADELTA: An adaptive learning rate method. arXiv 2012. arXiv preprint arXiv:1212.5701, 1212.