CS230

Presidential Debate Cross-talk Separation With U-Nets

Thatcher Freeman
Department of Computer Science
Stanford University
tfr@stanford.edu

Nick Myerberg
Department of Computer Science
Stanford University
nrm@stanford.edu

Danny Schwartz
Department of Computer Science
Stanford University
deschwa2@stanford.edu

Abstract

Neural networks have been shown to be useful in the task of audio source segmentation – the task of separating a mixed audio clip into its constituent tracks, such as turning a song into separate vocal and instrumental components. We apply this technique towards a real-world, practical task: separating the voices of politicians who are speaking on top of each other. We propose a new loss function that facilitates learning for this task.

1 Introduction

On September 29, 2020, Fox News hosted the US 2020 presidential election's first debate between President Donald Trump and President-Elect Joe Biden. Unlike many historic American presidential debates, this debate was plagued with frequent interruptions between the candidates, making it difficult for the viewers to hear the candidates' answers to questions. This prompted the Commission on Presidential Debates to take the unprecedented step of changing the debate format in an attempt to prevent interruptions.

Using deep learning to separate the muddled debate audio would relieve a lot of the frustrations of the viewers. How well could a deep learning approach perform in isolating Biden's voice when the human listeners had trouble with it? We decided to construct a model to solve this problem.

The task of separating concurrent speakers is known as Cocktail-Party Source Separation (1). However, much of the research in the audio separation domain is instead about the very similar task of Singing Voice Separation. In this task, the objective is to isolate the vocal and instrumental tracks of an input song, something that would certainly be useful for Karaoke or making remixes of music.

Learning to recognize and isolate or suppress an individual's voice has potential to be useful beyond screening hecklers, such as focusing on the voice of an instructor or a loved one. Speech isolation could also be useful if integrated with technology like hearing aids.

For both the Cocktail-Party Source Separation task and the Singing Voice Separation task, the technique is almost identical. The input is a mixed audio clip: a single channel of audio which contains a weighted sum of several audio sources. This mixed audio clip is divided into short segments that are a few seconds long. For each segment, we create a spectrogram, which contains the magnitudes of the Short-Time Fourier Transform (STFT) of the audio segment. Magnitudes are used to avoid operating on the STFT's complex outputs. The spectrogram is therefore a real 2D matrix that represents the intensities of different frequencies for different times in the audio segment. This spectrogram is passed into a neural network, which outputs a mask, a 2D matrix of the same shape as the spectrogram with values between 0 and 1. This mask is element-wise multiplied by the STFT of the input audio segment, and the result can be inverted (via an inverse STFT) to obtain an isolated source from that audio segment. This technique was first demonstrated in (1).

We build our baseline results by applying the U-Net network architecture used in (2), which is similar to that of (3). The core design is a convolutional auto-encoder that includes skip connections from the encoder stages to the decoder stages. This will be covered in more detail in section 4.1. [1]

[1] The model's source code is available at https://github.com/thatcherfreeman/CS230-FinalProject

CS230: Deep Learning, Fall 2020, Stanford University, CA.

2 Related Work

One of the first papers to approach the Cocktail-Party Source Separation task was (1). Simpson describes high quality results, but they were in ideal circumstances – the speakers were known male and female individuals, speaking in a controlled environment and potentially with very different voice pitches that make the sources easier to separate. Additionally, to minimize the size of his fully connected layers, Simpson downsamples the audio to a low sampling rate of 4 kHz which will inherently degrade the quality of the results. We found that downsampling an audio file below 12 kHz results in a significant degradation in vocal clarity.

Other work such as (2) uses a similar technique to preprocess audio clips. However, Jansson et al. instead use a fully convolutional U-Net, allowing them to process larger inputs than Simpson could in (1). We adopt our training procedure and architecture from the work in (2).

3 Dataset

We collected audio clips of Donald Trump and Joe Biden talking from YouTube, in the form of interviews and speeches. These audio clips were converted to WAV files, downsampled to 14.7 kHz, then subdivided into many partially overlapping 4.5 second clips. In total, we prepared 3014 unique clips of Biden (about 3.88 hours of audio) and 3475 unique clips of Trump (about 4.68 hours of audio). With this collection of 4.5 second Biden clips and 4.5 second Trump clips, we used peak-to-peak normalization on each clip and superimposed their audio together through a linear sum. Then, we took the short-time-frequency-transforms (STFT) of the combined audio and the Biden audio. To obtain an STFT with 128 time-windows of 512 different frequencies, we used Hanning windows of 1022 samples with an overlap of 511 samples. The resulting outputs were 512×128 arrays of complex numbers. Taking the element-wise magnitude of these STFTs results in a spectrogram. This STFT preprocessing approach was described in (1).

Our training and validation dataset consisted of three types of data. For 15,000 random unique pairings of normalized Biden and Trump segments, we took the linear sum to create a synthetic Combined audio clip. The Combined audio clips were converted to 512×128 spectrograms that would serve as the model's input. For each combined audio spectrogram X, we create a spectrogram for the corresponding Biden audio clip Y_{Biden} and the corresponding Trump audio clip Y_{Trump}, and these serve as the labels for X. We generated 15,000 unique examples of X, Y_{Biden}, Y_{Trump} and randomly chose 10% to be our validation dataset, saving the rest for training.

4 Method

4.1 Architecture

We followed in the steps of (2) in choosing our model architecture. The model consists of six encoders followed by six decoders. Each encoder consists of a single convolutional layer with 'replicate' padding of 2, a 5×5 kernel and with stride 2, creating an output that halves the input resolution in both dimensions. The first encoder has 16 filters and the following ones all have twice as many filters as the previous. Each CONV layer is followed by a leaky ReLU activation function with 0.2 leakiness.

The six decoders consist of a transposed convolution layer with window size 5×5 and stride 2, with 2-padding to effectively double the height and width of the input. The number of filters in the transposed convolution is equal to half of its input in all cases, except for the output decoder where the number of filters is 1. The output of the transposed convolution is concatenated with the output of the corresponding encoder layer along the channel dimension, and this tensor is sent to a series of two convolutional layers with window size 3×3 and stride 1, with 1-padding. These two convolutional layers had the same number of filters. ReLU was used after each of the transposed convolution and the convolution layers in the decoder. The three decoders with the most channels had a dropout of 0.5. By default, our final activation function was a sigmoid. See 1 for a diagram.

It was ambiguous how many convolutional layers were in the decoders in (2), but we observed that using two such layers after the concatenation resulted in improved performance over a single layer.

4.2 Masks

The most commonly used approach to source separation tasks is to use a neural network f to generate a mask, $f(X)$, such that the element-wise multiplication $f(X) * X$ is close to some target Y. The mask may also be thresholded, setting each element to zero or one if the value exceeds some α. The motivation for this method is twofold. First is that it creates a simple task for the network, treating the output as a per-pixel binary classification problem. The second benefit of this

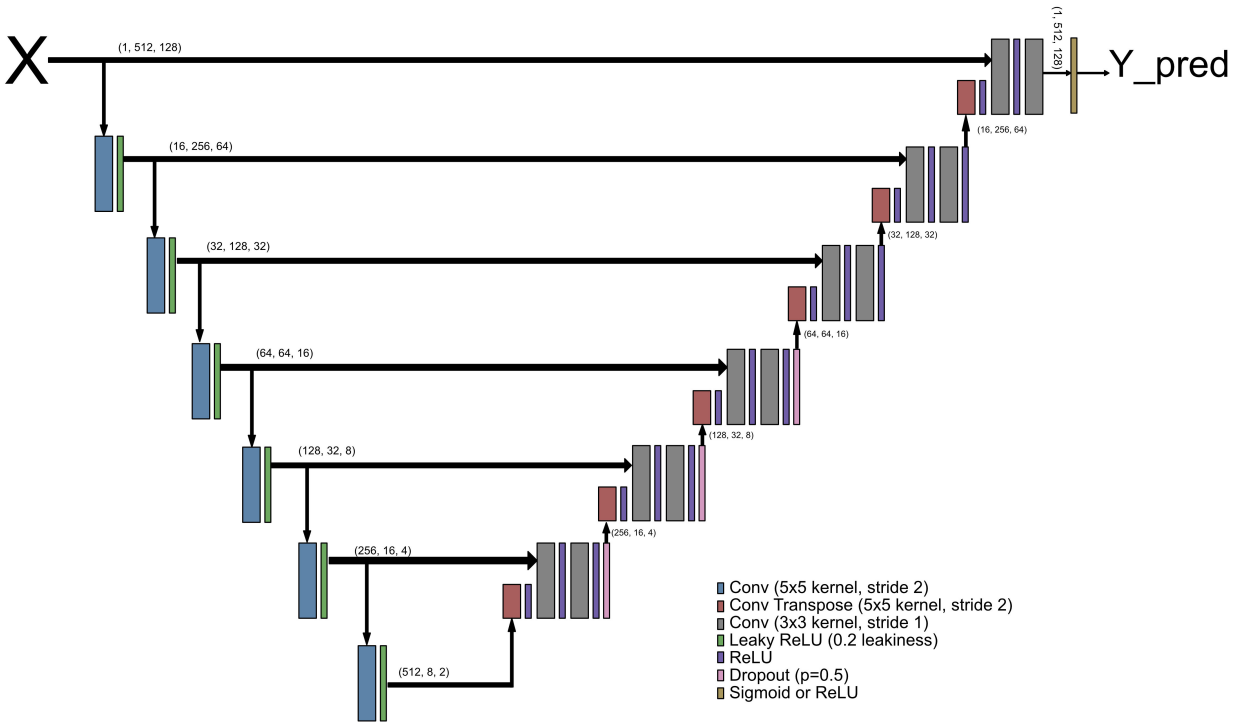

Figure 1: Model Architecture, arrows are labeled with the dimension of the activation that flows through it.

method is practical: most neural network libraries only support computations on real numbers, so this avoids computations on the complex STFT of the combined audio data.

4.3 Loss Function

We initially followed the approach used by (1) and (2), which gave us the following loss function:

$$\mathcal{L} = \texttt{mean}\left(|X * f(X) - Y_{Biden}|\right) \tag{1}$$

Above, $f(X)$ is the model's predicted mask, so the output activation function of the model is a sigmoid. One notable consequent of this loss function is that in regions of the spectrogram where X is close to zero, the product $X * f(X)$ effectively zeros out the gradients in those parts of the mask. This reduces the gradients corresponding to those parts of the spectrogram, making it difficult for the model to learn from regions of the spectrogram that are small. Effectively, this loss function encodes an assumption that we do not care what value the model chooses for the mask in low magnitude regions of the spectrogram.

To overcome this shortcoming, we propose a different loss function that incentivizes the model to learn the specific parts of the spectrogram that correspond to a source. Instead of trying to directly estimate a mask, we use $f(X)$ to estimate the magnitudes of the target, turning the task into an element-wise regression problem. In doing so, we change the output activation function of the model to ReLU, observing that every entry in a spectrogram is non-negative. Thus, our alternative loss function is:

$$\mathcal{L} = \texttt{mean}\left(|f(X) - Y_{Biden}|\right) \tag{2}$$

We can create a mask by taking the element-wise quotient $f(X)/X$, and clamp this matrix to the 0-1 range. This can be multiplied by the STFT corresponding to X, just as before.

Note that in both equations 1 and 2, we are taking the mean absolute difference between the predicted magnitudes and the true magnitudes, but in 1, the predicted magnitudes are generated by multiplying the model's predicted mask by X and in 2,

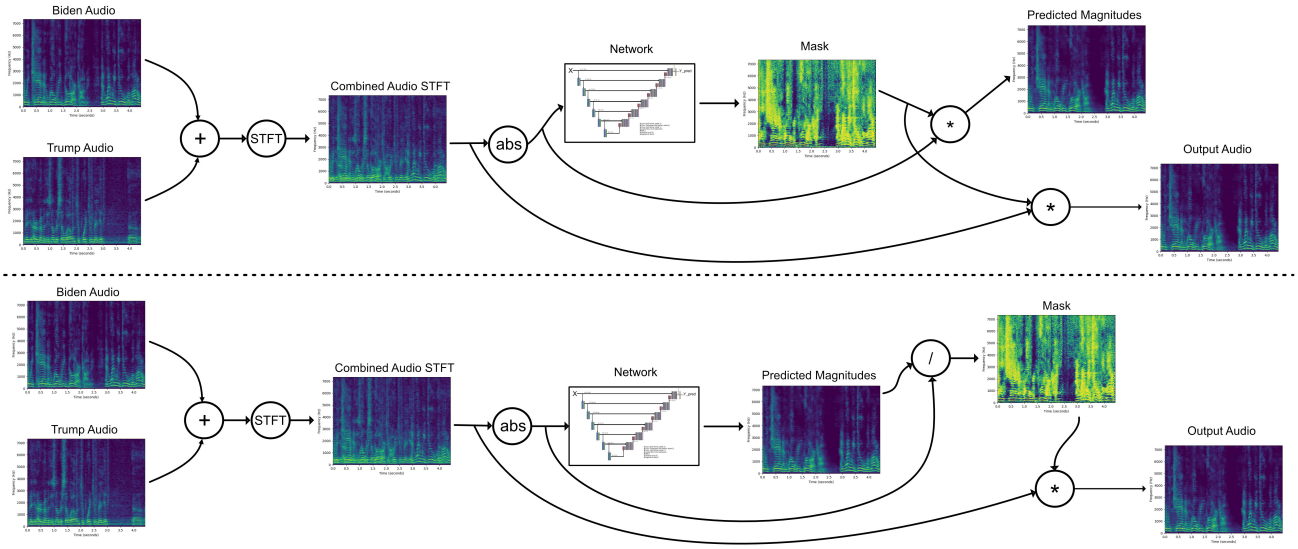

Figure 2: The upper half of this diagram represents how the model is used with the first loss function 1 and the lower half depicts usage of the model with our proposed loss function 2. In either case, the loss function minimizes the difference between the Predicted Magnitudes and the target audio source's magnitudes (Biden Audio), but the standard loss function 1 backpropagates this difference through the original audio.

Trial Number	1	2	3	4	5
Loss 1	8.376	8.301	8.687	17.133	17.132
Loss 2	7.197	8.159	6.657	6.836	7.258

Table 1: Validation loss from training the U-Net over five trials (mean absolute difference between predicted magnitude and ground truth spectrogram). Lower is better.

the predicted magnitudes are directly learned by the model. Refer to 2 for how inference works with these two variants of the model.

5 Training

We used the Adam optimizer with a 1e-4 learning rate and trained for 30 epochs with a batch size of 5. L2 regularization did not appear to improve validation accuracy, so our only form of regularization was in the three dropout layers ($p = 0.5$). The learning rate and batch size were selected to maximize training speed in our initial experiments and the dropout probability was chosen to follow (2). The learning rate was unchanged between the two loss functions.

6 Results

6.1 Comparison of loss functions

We observed that training with the first loss function 1 was inconsistent. Sometimes it would train successfully and other times the mask would simply converge to all zeros or all ones. This appeared to occur at random, on consecutive runs, suggesting that this was dependent on the random initialization. However, with 2, training was consistent, our model converging more quickly, and our loss function achieved better accuracy on our validation set (computed by the mean absolute difference between the masked input and the target output).

To show this empirically, we trained our models five times with each loss function and plotted the results in Table 1. We obsesrved similar results in the training process—the first loss function failed to optimize a substantial fraction of the time, with loss not improving over the course of the 30 epochs.

6.2 Qualitative Results

There are a variety of possible post-processing methods that we could use with these models. The one used in (1) and (2) was to generate the mask, and then push entries that exceeded some α to 1, otherwise setting those values to 0. One benefit of this method is that it fully mutes parts of the spectrogram that the model predicted do not belong to a given source. However, it also has the consequence of introducing audible and distracting artifacts in regions where the model was imprecise.

Another option is to simply use the mask generated by the model as-is. This reduces the intensity of the source separation, but typically results in more pleasant results.

Despite being trained almost entirely on cross-talk samples, the model generalized somewhat well to audio clips that only contained Trump or only contained Biden speaking, properly muffling Trump and leaving Biden untouched in our validation set.

6.3 Quantitative Results

One of the most popular metrics in source separation tasks is the Source to Distortion Ratio, as defined by (4). This metric is the log ratio between the source correlation of the predicted audio and the ground truth audio, and the distortion (artifacts, source interference, and background noise) present in the predicted audio track. With our model trained against the baseline loss function 1, we were able to attain an SDR of 7.633. With our model trained with the second loss function 2, we attained an SDR of 8.410 on our validation data. We used an implementation of this metric available at `https://github.com/sigsep/bsseval`

7 Future Work

One major drawback to this approach in cross-talk source separation is that the model is trained to only separate the voices of two specific people. One possible future approach would be to overlay a target individual's voice with many other speakers, so the model can be trained to learn to isolate that specific person's voice and generalize to more use cases. Another aspect worth exploring would be the usage of wavelet transforms instead of STFTs to convert the audio into a more machine-readable format.

8 Conclusion

We looked at applying the methods used in (1) and (2) towards a more difficult task: source separation when both sources are elderly men. In the process, we propose a modification to the previous training methods that more directly optimizes the model's understanding of the sources, and results in improved training characteristics.

9 Contributions

9.1 Danny

Danny wrote all of the audio processing code we used to prepare our dataset. He also gathered and hand-edited some training data. Danny collaborated with the team to iterate on the model and our objective function. He also collaborated with the team during our literature review process.

9.2 Thatcher

Thatcher wrote the PyTorch code that models the Neural Network, the training and test loops, and made the diagrams in this writeup and the video. He was also responsible for training the models and creating the proposed loss function.

9.3 Nick

Nick was responsible for data collection, identifying, converting, and editing (removal of other speakers) Biden and Trump speeches and interviews to useful audio files, in preparation for the data pre-processing phase. He also handled the infrastructure part of the project, managing the AWS instance and billing.

References

[1] A. J. Simpson, "Probabilistic binary-mask cocktail-party source separation in a convolutional deep neural network," *arXiv preprint arXiv:1503.06962*, 2015.

[2] A. Jansson, E. Humphrey, N. Montecchio, R. Bittner, A. Kumar, and T. Weyde, "Singing voice separation with deep u-net convolutional networks," 2017.

[3] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[4] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE transactions on audio, speech, and language processing*, vol. 14, no. 4, pp. 1462–1469, 2006.