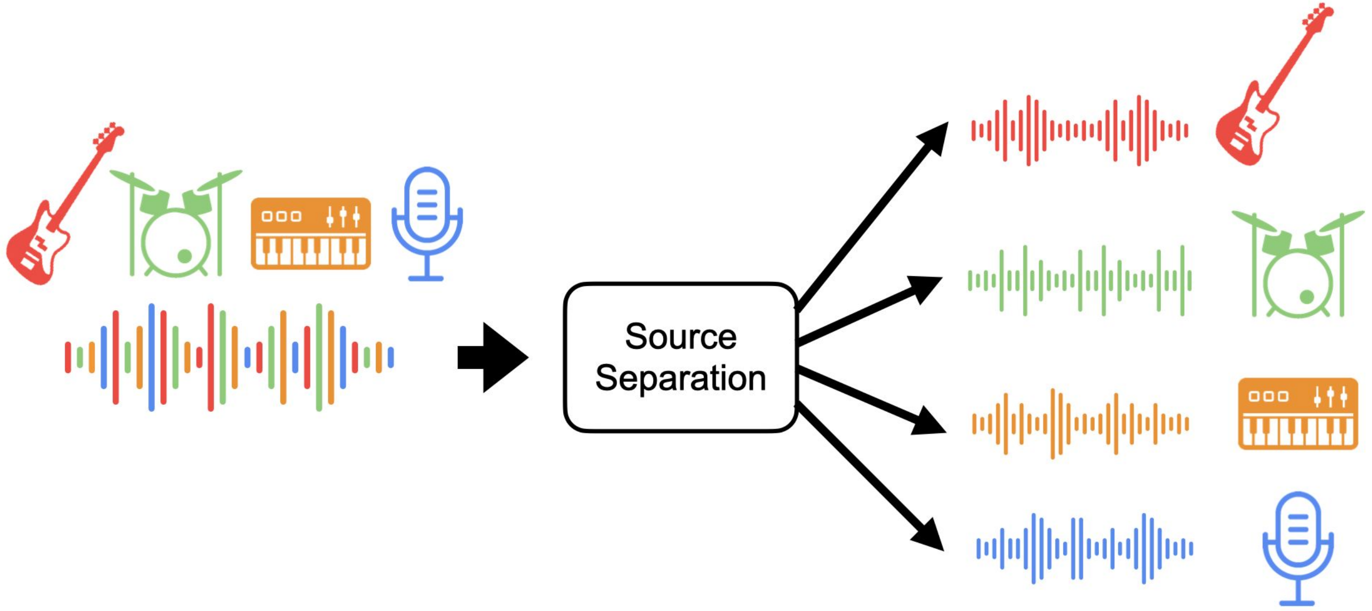
Audio Source Separation

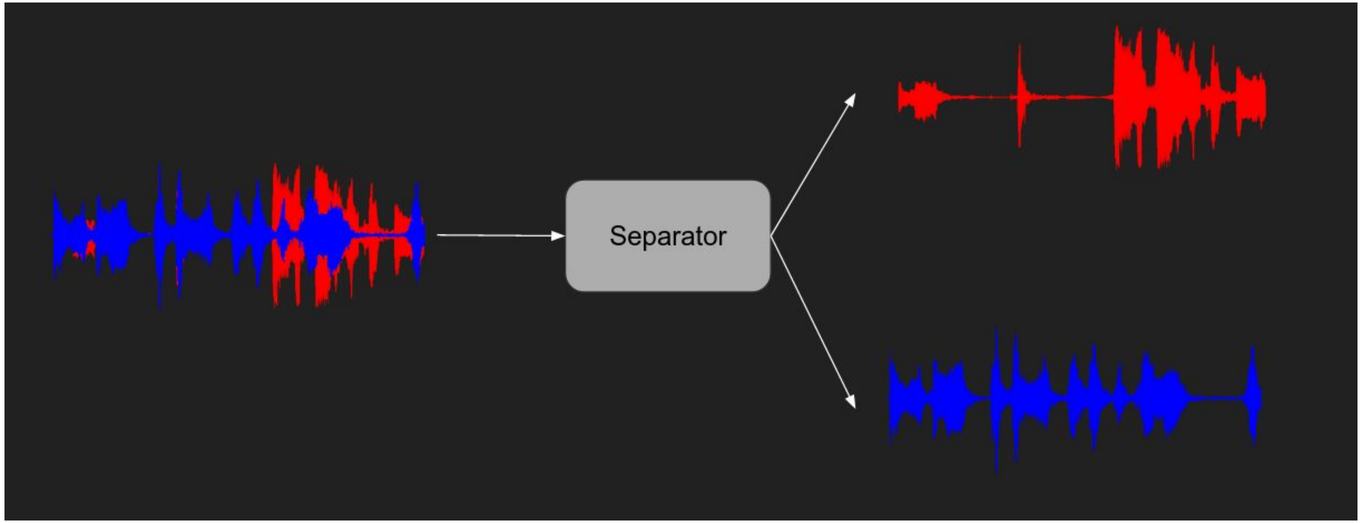
EE698R course project
[Conscious?]
(Harsh Kumar , Shorya Kumar)

Introduction

Audio Source Separation



What we are doing

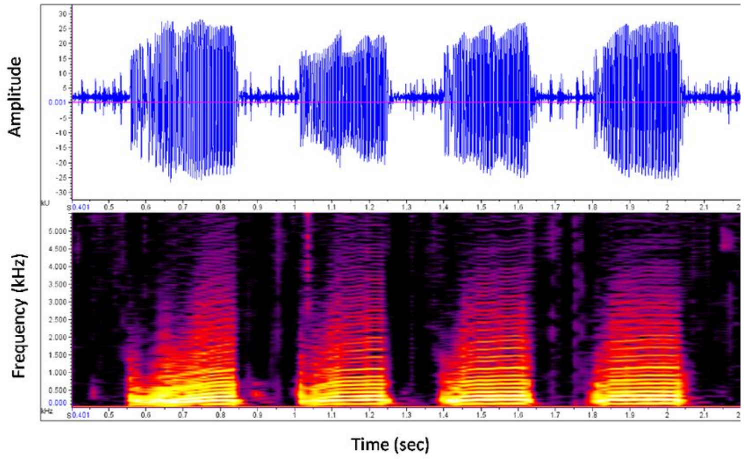


Source: https://medium.com/analytics-vidhya/improving-voice-separation-by-incorporating-end-to-end-speech-recognition-c3dd57d80206

Paper Approach

Frequency domain v/s Time domain

- Models for separation usually operate on magnitude spectrum, which ignores phase information.
- The **STFT** output depends on many parameters, such as the **size** and **overlap of audio frames**, which can affect the time and frequency resolution. Ideally, these parameters should be optimised with parameters of the separation model to maximise performance for a particular separation task. In practice, however, the transform parameters are fixed to specific values.



https://www.researchgate.net/figure/Spectrograms-and-Oscillograms-This-is-an-oscillogram-and-spectrogram-of-the-boatwhistle_fig2_267827408

Source: https://arxiv.org/pdf/1806.03185.pdf [WAVE-U-NET: A MULTI-SCALE NEURAL NETWORK FOR END-TO-END AUDIO SOURCE SEPARATION]

Frequency domain v/s Time domain

- Secondly, the separation model does not estimate the **source phase**, it is often assumed to be equal to the **mixture** phase, which is incorrect for overlapping partials. Alternatively, the **Griffin-Lim algorithm** can be applied to find an approximation to a signal whose magnitudes are equal to estimated ones, but this is slow and often no such signal exists.

- Lastly, the mixture phase is ignored in the estimation of sources, which can potentially limit the performance. Thus, it would be desirable for the separation model to learn to estimate the source signals including their phase directly

Source: https://arxiv.org/pdf/1806.03185.pdf [WAVE-U-NET: A MULTI-SCALE NEURAL NETWORK FOR END-TO-END AUDIO SOURCE SEPARATION]

Data Preprocessing

MIR-1k dataset

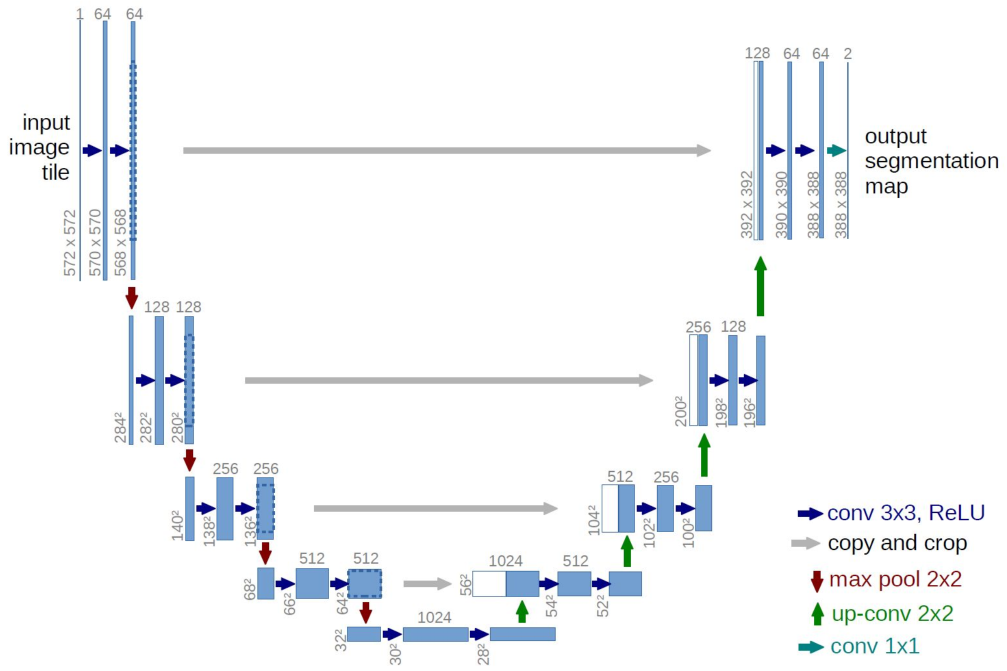
1. **1000** song clips which the **music accompaniment** and the **singing voice** are recorded at **left** and **right** channels, respectively.
2. **Manual annotations** of the dataset include pitch contours in semitone, indices and types for unvoiced frames, lyrics, and vocal/non-vocal segment.
3. The speech recordings of the lyrics by the same person who sang the songs are also provided in the dataset.

Our Modifications

1. We used only the undivided 2-channel audio files present in MIR-1K, and created the input and output dataset by **merging** the channels and **separating** them respectively.
2. Hence our model takes a **1-channel audio (Mono)** file as input and gives 2 **1-channel audio (Mono)** files, one for the **music accompaniment** and one for the **vocals**.
3. The audio files had lengths varying from 22 secs to 126 secs.
4. We clipped all audio files to 1 sec audio files (sampled at 16kHz) due to machine specifications and constraints. Thus we got around **7947** 1-sec samples.

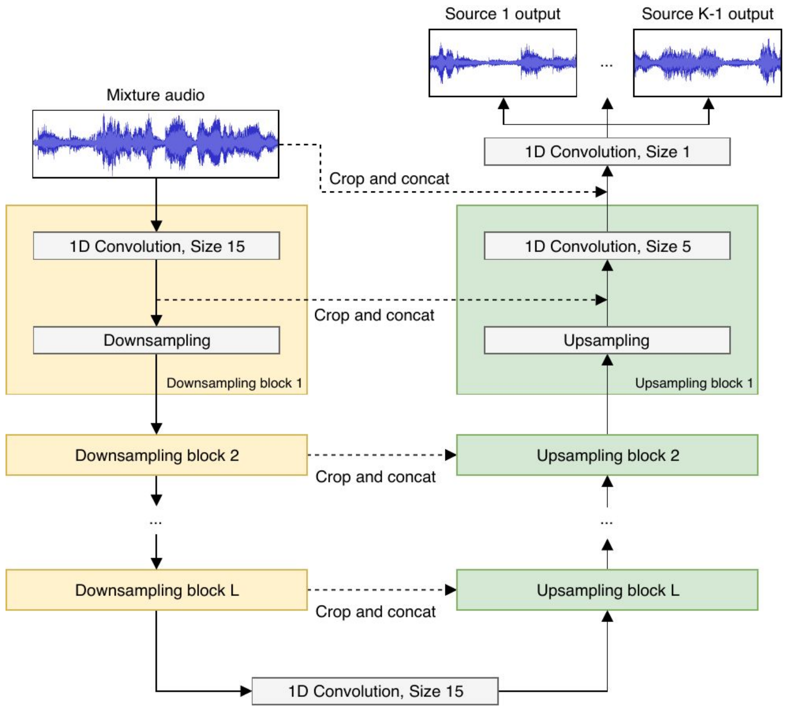
Model summary

U-Net Model

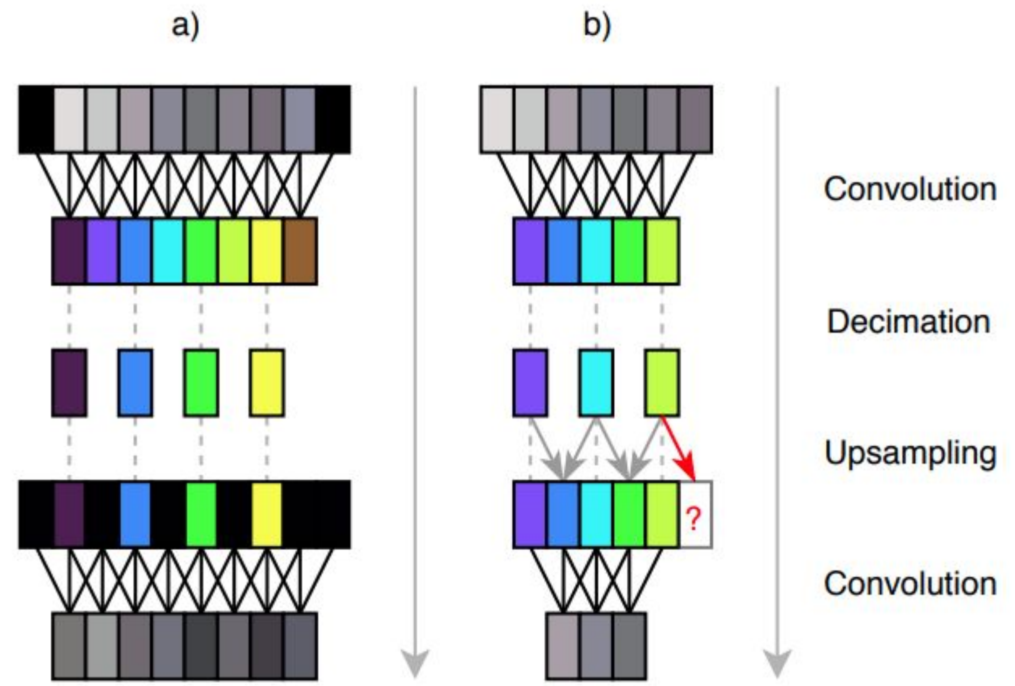


Source: https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/

Wave U-Net Model

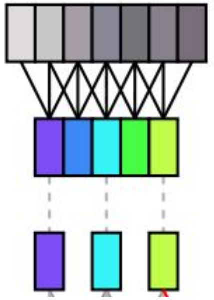


Source: https://arxiv.org/pdf/1806.03185.pdf [WAVE-U-NET: A MULTI-SCALE NEURAL NETWORK FOR END-TO-END AUDIO SOURCE SEPARATION]



Source: https://arxiv.org/pdf/1806.03185.pdf [WAVE-U-NET: A MULTI-SCALE NEURAL NETWORK FOR END-TO-END AUDIO SOURCE SEPARATION]

Downsampling Block

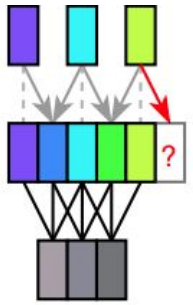


Source: https://arxiv.org/pdf/1806.03185.pdf

- **Convolution**
  This is a general **1D convolutional layer** with a **fixed kernel size** and **fixed number of filters** for each block individually. We take a **stride of 1**, hence the output of this layer would be ***L - k + 1*** where ***L*** is the initial length of the 1D input to this layer and ***k*** is the kernel size. This follows no padding and an activation layer of ***Leaky-ReLU***.

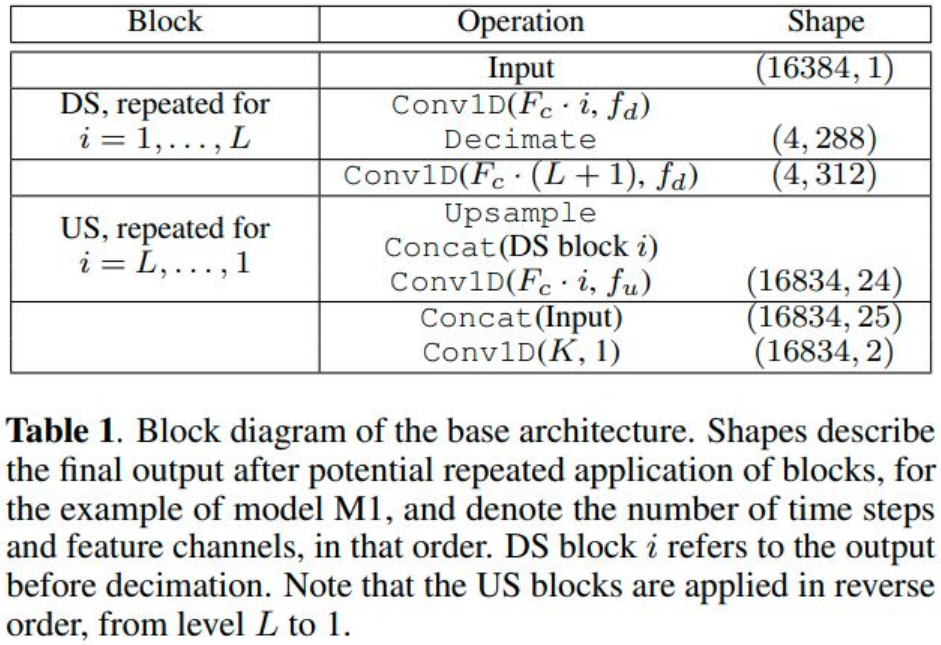
- **Downsampling**
  Here the resolution of the input, given to this layer, is **halved** by dropping out every **alternate** frame/element of the input and hence the output length being **half** the input length.

Upsampling Block



Source: https://arxiv.org/pdf/1806.03185.pdf

- **Upsampling**
  Here the resolution of the input is almost **doubled** by **expanding** the input into **twice its length** and filling the gaps with the **mean of the values of their neighbours**. Note that this is NOT done for the first and last value. So, a ***L*** length input becomes a ***2L - 1*** length output.
- **Crop and Concat**
  For the i^{th} upsampling block, the output of the convolutional layer of the i^{th} downsampling block is **cropped** to fit the input of this layer and then **concatenated** on the input along the channels.
- **Convolution**
  This is a general **1D convolutional layer** similar to the one used in the downsampling block.



Block	Operation	Shape
	Input	$(16384, 1)$
DS, repeated for $i = 1, \ldots, L$	$\texttt{Conv1D}(F_c \cdot i,\ f_d)$ $\texttt{Decimate}$	 $(4, 288)$
	$\texttt{Conv1D}(F_c \cdot (L+1),\ f_d)$	$(4, 312)$
US, repeated for $i = L, \ldots, 1$	$\texttt{Upsample}$ $\texttt{Concat}$(DS block i) $\texttt{Conv1D}(F_c \cdot i,\ f_u)$	 $(16834, 24)$
	$\texttt{Concat}$(Input) $\texttt{Conv1D}(K, 1)$	$(16834, 25)$ $(16834, 2)$

Table 1. Block diagram of the base architecture. Shapes describe the final output after potential repeated application of blocks, for the example of model M1, and denote the number of time steps and feature channels, in that order. DS block i refers to the output before decimation. Note that the US blocks are applied in reverse order, from level L to 1.

Source: https://arxiv.org/pdf/1806.03185.pdf [WAVE-U-NET: A MULTI-SCALE NEURAL NETWORK FOR END-TO-END AUDIO SOURCE SEPARATION]

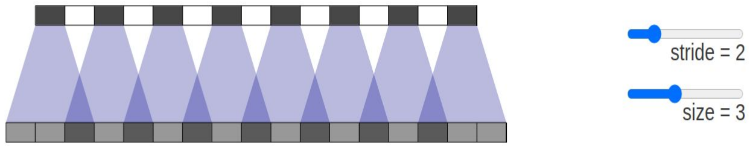
Training

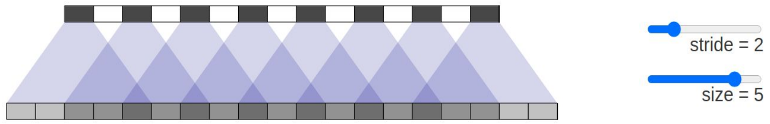
- **Loss Function** - Mean Squared Error (MSE)
- **Optimizer** - Adaptive Moment (AdaM) Optimizer

  With a learning rate of $\eta = 0.001$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$
- **Batch Size** - 16
- **Input Length** - 16000 (as audio is sampled at 16kHz)
- **No of Layers, *L*** - 11
- **Extra Filters per layer, F_c** - 24
- **Filter Size/Kernel Size** - 15 for downsampling block and 5 for upsampling block

Question from last time :
Why not Transposed Convolution

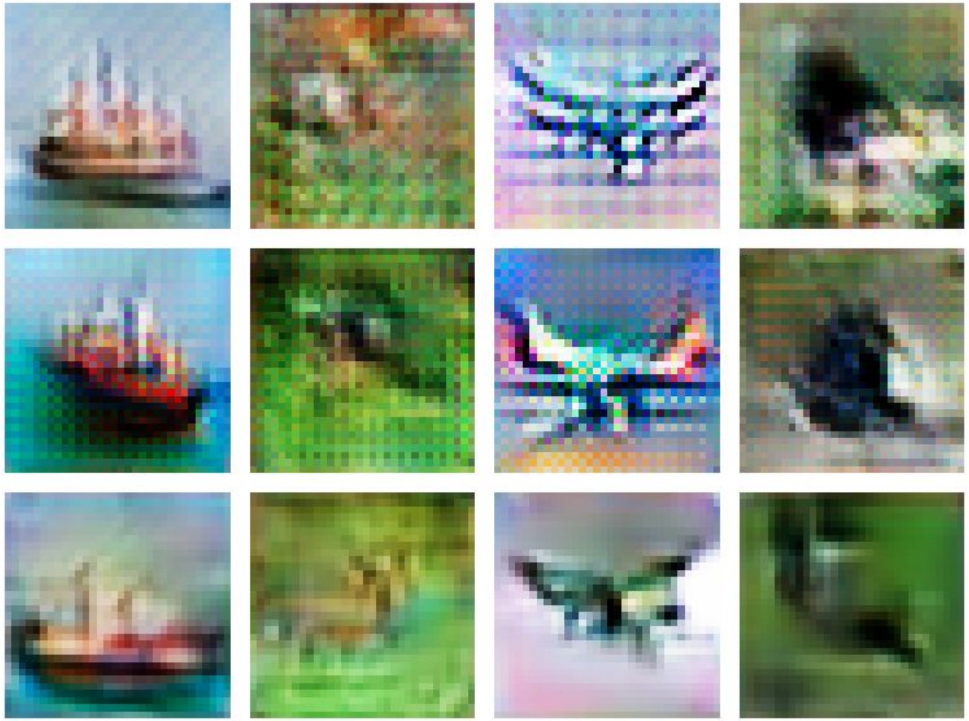
- Upsampling artifacts are caused by problematic upsampling layers and due to spectral replicas that emerge while upsampling. Also, depending on the used upsampling layer, such artifacts can either be tonal artifacts (additive high-frequency noise) or filtering artifacts (subtractive, attenuating some bands)
- Unfortunately, deconvolution can easily have “uneven overlap,” putting more of the metaphorical paint in some places than others . In particular, deconvolution has uneven overlap when the kernel size (the output window size) is not divisible by the stride (the spacing between points on the top). While the network could, in principle, carefully learn weights to avoid this, in practice neural networks struggle to avoid it completely.





- When such convolutions were used as upsampling blocks in the Wave-U-Net model, artifacts were found in the form of high-frequency buzzing noise.

Source: https://distill.pub/2016/deconv-checkerboard/ , https://arxiv.org/pdf/2111.11773.pdf



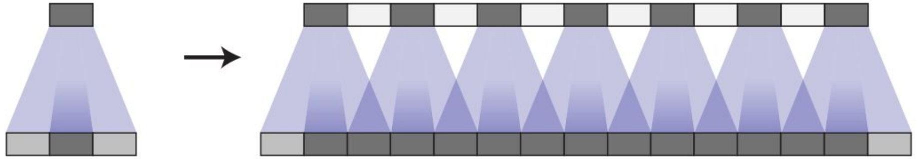
Deconv in last two layers.
Other layers use resize-convolution.
Artifacts of frequency 2 and 4.

Deconv only in last layer.
Other layers use resize-convolution.
Artifacts of frequency 2.

All layers use resize-convolution.
No artifacts.

Source: https://distill.pub/2016/deconv-checkerboard/ , https://arxiv.org/pdf/2111.11773.pdf

- Thinking about things in terms of uneven overlap is — while a useful framing — kind of simplistic. For better or worse, the model learns weights for their deconvolutions. In theory, our models could learn to carefully write to unevenly overlapping positions so that the output is evenly balanced.



- In fact, not only do models with uneven overlap not learn to avoid this, but models with even overlap often learn kernels that cause similar artifacts! While it isn't their default behavior the way it is for uneven overlap, it's still very easy for even overlap deconvolution to cause artifacts.

Source: https://distill.pub/2016/deconv-checkerboard/ , https://arxiv.org/pdf/2111.11773.pdf

Architectural Improvements

Difference Output Layer

The baseline model outputs **one source estimate** for each of K sources by **independently** applying **K convolutional filters** followed by a *tanh* nonlinearity to the last feature map.

We consider the mixture signal as the **sum of its source signal components**:

$$\mathbf{M \approx S_1+S_2+\ldots+S_K.}$$

Since the baseline model is **not constrained** in this fashion, it has to learn this rule approximately to avoid highly improbable outputs, which could **slow down learning** and **reduce performance**.

Therefore, the authors use a **difference output layer** to constrain the outputs $\mathbf{S_j}$ such that only ***K - 1*** source signals are estimated and the last source is then simply computed as

$$\mathbf{S_K \approx M - (S_1+S_2+\ldots+S_K).}$$

Source: https://arxiv.org/pdf/1806.03185.pdf [WAVE-U-NET: A MULTI-SCALE NEURAL NETWORK FOR END-TO-END AUDIO SOURCE SEPARATION]

Learned Upsampling

Linear interpolation for upsampling is simple, **parameter-less** and encourages **feature continuity**. However, it may be restricting the network capacity too much.

The authors have proposed the **learned upsampling layer**. For a given ***F × n*** feature map with ***n*** time steps, we compute an interpolated feature $\mathbf{f}_{t+0.5} \in \mathbf{R}^F$ for pairs of neighbouring features $\mathbf{f}_t$, $\mathbf{f}_{t+1} \in \mathbf{R}^F$ using parameters $\mathbf{w} \in \mathbf{R}^F$ and the sigmoid function $\boldsymbol{\sigma}$ to constrain each $\mathbf{w}_i \in \mathbf{w}$ to the [0, 1] interval:

$$\mathbf{f}_{t+0.5} = \sigma(\mathbf{w}) \square \mathbf{f}_t + (1 - \sigma(\mathbf{w})) \square \mathbf{f}_{t+1}$$

The learned interpolation layer can be viewed as a **generalisation of simple linear interpolation**, since it allows **convex combinations** of features with weights other than 0.5.

Source: https://arxiv.org/pdf/1806.03185.pdf [WAVE-U-NET: A MULTI-SCALE NEURAL NETWORK FOR END-TO-END AUDIO SOURCE SEPARATION]

Evaluation

SDR (Signal-to-Distortion Ratio)

- The **signal-to-distortion (SDR) metric** is commonly used to evaluate source separation performance . An audio track is usually partitioned into non-overlapping audio segments multiple seconds in length, and **segment-wise** metrics are then **averaged over each audio track or the whole dataset** to evaluate model performance.

$$\text{SDR} := 10 \log_{10} \left(\frac{\|s_{\text{target}}\|^2}{\|e_{\text{interf}} + e_{\text{noise}} + e_{\text{artif}}\|^2} \right)$$

- The SDR computation is problematic when the **true source is silent or near-silent**. In case of silence, the SDR is **log(0)**, i.e. undefined , which happens often for vocal tracks.

Source: https://arxiv.org/pdf/1806.03185.pdf [WAVE-U-NET: A MULTI-SCALE NEURAL NETWORK FOR END-TO-END AUDIO SOURCE SEPARATION]

MAD (Mean Absolute Deviation)

- Since the collection of **segment-wise vocal SDR** values across the dataset is not normally distributed , the **mean and standard deviation are not sufficient** . As a workaround, authors take the **median over segments**, as it is **robust against outliers** , using the median absolute deviation (MAD) as a **rank-based equivalent** to the standard deviation (SD).

- We also note that **increasing the duration of segments** beyond one second alleviates this issue by removing many, but not all outliers. This is more **memory-intensive** and presumably still **punishes errors** during silent sections most.

Source: https://arxiv.org/pdf/1806.03185.pdf [WAVE-U-NET: A MULTI-SCALE NEURAL NETWORK FOR END-TO-END AUDIO SOURCE SEPARATION]

Limitations

Inconsistencies in Boundaries

Since the model’s input and output are of **equal length** and the total output is created by **concatenating** predictions for non-overlapping consecutive audio segments, inconsistencies and discontinuities emerge at the **borders**.

The loudness might abruptly decreases at the end of a 4s, and a beginning vocal melisma is suddenly cut off at 2s seconds, leaving only quiet noise, before the vocals reappear at 4s.

Lost Temporal Context

A **vocal melisma** with only the vowel “a” can sound similar to a non-vocal instrument and presumably was mistaken for one because **no further temporal context was available**.

This model is less capable of performing separation wherever information from a temporal context is required. Larger input and output sizes somewhat alleviate the issue.

Source: https://arxiv.org/pdf/1806.03185.pdf [WAVE-U-NET: A MULTI-SCALE NEURAL NETWORK FOR END-TO-END AUDIO SOURCE SEPARATION]

Team

Harsh Kumar
(harshku@iik.ac.in)

Shorya Kumar
(shoryak@iik.ac.in)

Q/A

- Where is the paper published
- What did you do in the project exactly
- How every function works in training
- Importance of audio source separation
- Maths add kardena slides me
- Vanishing gradients problem in WaveUNet

Thank You!!!