# Audio Separation using Attention in Linear Time

Enrique Nueve

May 2020

## 1 Abstract

Transformers have been recently explored as an alternative means for the task of audio separation compared to using more traditional RNN based architectures. Methods using transformers for audio separation show a promising path forward given that they have achieved state-of-the-art (SOTA) results. However, due to the $O(N^2)$ memory complexity of performing Self-Attention within the transformer block where N is the length of the input source, the practical applications of such methods may be hindered relative to the hardware being used to run such a model. In this paper, we propose the RHA, a Transformer-based neural network for audio separation that uses an efficient version of Self-Attention that has an optimal memory complexity of $O(N)$. The proposed model achieves (WORD WORD WORD) performance on the FUSS and LibriMix datasets. Through the usage of a memory-efficient alternative to Self-Attention, we are able to achieve competitive performance with SOTA methods yet with a memory complexity of only $O(N)$ for our Transformer block.

## 2 Possible publication spots

1. (Aug 31) https://www.journals.elsevier.com/journal-of-parallel-and-distributed-computing/call-for-papers/distributed-intelligence-at-the-edge-for-the-future
2. (Aug 6) https://attend.ieee.org/ssci-2021/ieee-symposium-on-computational-intelligence-in-iot-and-smart-cities-ieee-ciiot/
3. (Aug 31) https://fcrlab.unime.it/calls/artificial-intelligence-in-the-edge
4. (Aug 31) https://www.journals.elsevier.com/journal-of-parallel-and-distributed-computing/call-for-papers/distributed-intelligence-at-the-edge-for-the-future
5. (July 10) https://www.journals.elsevier.com/computers-and-electrical-engineering/call-for-papers/artificial-intelligence-driven-mobile-edge-computing-vsi-aim
6. (Oct 01) https://2022.ieeeicassp.org/ , deadline October 01, 2021

## 3 Doing now :)

1. ...

## 4 To-do

1. prep LibriMix
2. Clip on loss
3. Add Dynamic Mixing
4. sanity check dimensions to make sure attention is applied to right time axis
5. work through more deviation for SPE
6. work through deviation for Low Rank Attention
7. merge all deviations

## 5 Literature Review

1. **Attention is All You Need in Speech Separation** [Sub+21]
   - **Link**: https://arxiv.org/abs/2010.13154
   - **Summary**:
   - **Assumptions**:
   - **Limits**:
   - **Potential Improvements**:
   - **What ideas will I use**:
2. **Attention is All You Need** [Vas+17]
   - **Link**: https://arxiv.org/abs/1706.03762
   - **Summary**:
   - **Assumptions**:
   - **Limits**:
   - **Potential Improvements**:
   - **What ideas will I use**:
3. **Rethinking Attention with Performers**[Cho+21]
   - **Link**: https://arxiv.org/abs/2009.14794
   - **Summary**:
   - **Assumptions**:
   - **Limits**:
   - **Potential Improvements**:
   - **What ideas will I use**:
4. **Reformer: The Efficient Transformer**[KKL20]
   - **Link**: https://arxiv.org/abs/2001.04451
   - **Code**: https://github.com/lucidrains/reformer-pytorch
   - **Summary**:
   - **Assumptions**:
   - **Limits**:
   - **Potential Improvements**:
   - **What ideas will I use**:
5. **Funnel Transformer** [Dai+20]
   - **Link**: https://arxiv.org/abs/2006.03236
   - **Summary**:

- **Assumptions**:
- **Limits**:
- **Potential Improvements**:
- **What ideas will I use**:

6. **Adaptive Attention Span**[Suk+19]
   - **Link**: https://arxiv.org/abs/1905.07799
   - **Summary**:
   - **Assumptions**:
   - **Limits**:
   - **Potential Improvements**:
   - **What ideas will I use**:
7. **Compresive Transformer**[Rae+19]
   - **Link**: https://arxiv.org/abs/1911.05507
   - **Summary**:
   - **Assumptions**:
   - **Limits**:
   - **Potential Improvements**:
   - **What ideas will I use**:
8. **The Evolved Transformer** [SLL19]
   - **Link**: https://arxiv.org/abs/1901.11117
   - **Summary**:
   - **Assumptions**:
   - **Limits**:
   - **Potential Improvements**:
   - **What ideas will I use**:
9. **Perceiver: General Perception with Iterative Attention** [Jae+21]
   - **Link**: https://arxiv.org/abs/2103.03206
   - **Summary**:
   - **Assumptions**:
   - **Limits**:
   - **Potential Improvements**:
   - **What ideas will I use**:
10. **SDR – HALF-BAKED OR WELL DONE?**

- **Link**: https://arxiv.org/pdf/1811.02508.pdf
- **Summary**:Explains loss function SI-SDR
- **Assumptions**:
- **Limits**:
- **Potential Improvements**:
- **What ideas will I use**: will use in citation in methodology for loss function.

## 6 Data

### 6.1 Fuss

- 8.89 GB to 25 gb
- https://zenodo.org/record/3694384.YPsbaRNKgq1
- Fuss ssdata, https://zenodo.org/record/3694384/files/FUSS_ssdata.tar.gz?download=1

### 6.2 Librimix

- https://github.com/JorisCos/LibriMix

# 7 Background

## 7.1 Self-Attention, Multi-Head Attention, and Transformer Blocks

Self-Attention allows for a map to be learned over a sequence where each element is weighted relative to every other element in the sequence. This allows the map to learn long-term dependencies between elements. Self-Attention take in an input $X \in \mathbb{R}^{S \times D}$ and applies three different linear weight layers: $W^Q \in \mathbb{R}^{D \times d_k}, W^K \in \mathbb{R}^{D \times d_k}$, and $W^V \in \mathbb{R}^{D \times d_v}$ to $X$. In practice, $d_k$ is often set equal to $d_v$. This produces queries $Q = XW^Q$, keys $K = XW^K$, and values $V = XW^V$.

$$Self\text{-}Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

By performing $QK^T$, Self-Attention is able to compute a similarity like operation between the quires and keys facilitating the learning of the dependencies between elements of the input sequence. Softmax is applied to $QK^T$ to form a weighting which is then applied to $V$ forming the output of the Self-Attention which can be thought of as a new version of the input $X$ in which now each element has embedded information about its neighbors within the original sequence $X$.

Multi-Head Attention (MHA) is a composition of multiple Self-Attention units. As stated in [Vas+17], MHA allows the model to jointly attend to information from different representation subspaces at different positions. MHA is formulated as shown below.

$$MHA = Multi\text{-}HeadAttention = Concat(head_1, head_2, \ldots, head_n)W^o$$
$$head_i = Self\text{-}Attention(Q_i, K_i, V_i) = softmax(\frac{Q_i K_i^T}{\sqrt{d_k}})V_i$$

Each $head_i$ has it's own set of weights for queries $W_i^Q$, keys $W_i^K$, and values $W_i^V$. In the original work [Vas+17], $d_k = d_v = \frac{D}{h}$ (h being the number of heads) and $W^o \in \mathbb{R}^{D \times D}$ which causes the output to have the original dimensionality of $S \times D$.

MHA is the key component used within a unit of layers called a Transformer block. The Transformer block consist of a MHA which takes in the input, a recurrent connection between the MHA output and the original input, and then a layer norm. After that, the previous output is feed through a dense layer, a recurrent connection is applied again, concluding with a layer norm. The Transformer block can be expressed by the formulation below where $LN$ is a layer norm , $D$ is a dense layer, and $X$ is the original input.

$$Transformer(X) = \mathrm{LN}(\mathrm{D}(\mathrm{LN}(\mathrm{MHA}(X) + X)) + \mathrm{LN}(\mathrm{MHA}(X) + X))$$

Before inputting $X$ into the Transformer block as well as even Self-Attention or MHA, a positional embedding such as Sinusoidal Positional Embedding is applied to $X$ This is due to Self-Attention not being able to in-itself be able to be aware of relative position. For example, Self-Attention would know that a sequence (A,B,C) consists of elements A, B, and C yet wouldn't know that A comes before B and B comes before C, etc. Positioning of a sequence is essential information for task such as NLP, ordering of words in a sentence. [Vas+17] proposed the use of Sinusoidal Positional Embedding which is formulated below in which the positional embeddings $PE$ are embedd into $X$ by simple addition $X^{'} = PE + X$ where $d_{model}$ is equal to the inputs embedding dim $D$. Pos refers to the position and $i$ refers to the dimensions of PE.

$$PE_{(pos,2i)} = \sin{(pos/10000^{2i/d_{model}})}$$
$$PE_{(pos,2i+1)} = \cos{(pos/10000^{2i/d_{model}})}$$

## 7.2 SepFormer: Speech Separation using a Transformer Block

[Sub+21] proposes the SepFormer, a speech seperation model that uses a Transformer block.

## 7.3 Short comings of Self-Attention and Proposed Solutions

Self-Attention has allowed Deep Learning models to achieve high SOTA results on an array of task is areas such as NLP, Audio Processing, and Computer Vision. Unfortunately, the computational performance of Self-Attention is rather poor given a memory complexity of $O(N^2)$ due to the $softmax(QK^T)$ operation. In recent work, such as [Cho+21; KKL20; Dai+20; SLL19; Rae+19; Suk+19; Jae+21; Kat+20], have explored many different ways to reduce the memory complexity of Self-Attention as well as parameter usage while still maintaining the performance quality of traditional Self-Attention. In our work we use a formulation of Self-Attention proposed by [Kat+20].

In the work of [Kat+20], they generalize Self-Attention into an operation that just performs a similarity operation between $Q$ and $K$. [Kat+20]'s formulation of Abstract Attention $A$ is shown below for each row of the output $A_i$.

$$A_i = \frac{\sum_{j=1}^{N} sim(Q_i, K_j)V_j}{\sum_{N\ j=1} sim(Q_i, K_j)}$$

If the similarity operation is chosen to be softmax then, standard Self-Attention is formed. By assuming that the similarity operation is a kernel, $K(x, y) : \mathbb{R}^{2\times F} \to \mathbb{R}_+$, $A_i$ can be re-written using the kernel trick and the associative property of matrix multiplication into the form shown below.

$$\begin{aligned} A_i &= \frac{\sum_{j=1}^{N} \phi(Q_i)^T \phi(K_j) V_j}{\sum_{j=1}^{N} \phi(Q_i)^T \phi(K_j)} \\ &= \frac{\phi(Q_i)^T \sum_{j=1}^{N} \phi(K_j) V_j^T}{\phi(Q_i)^T \sum_{j=1}^{N} \phi(K_j)} \end{aligned}$$

In the form above, the numerator can be vectorized, $(\phi(Q)\phi(K)^T)V = \phi(Q)(\phi(K)^T V)$, where $\phi$ is applied row-wise to Q and K. With this fomulation, the memory complexity is $O(N)$ since we can compute $\sum_{j=1}^{N} \phi(K_j)V_j^T$ and $\sum_{j=1}^{N} \phi(K_j)$ once and reuse them for each query. In the work of [Kat+20], $\phi$ is chosen to be ReLU. Although the proposed Abstract Attention above is not the standard Self- Attention from [Vas+17], [Kat+20] shows through experiments nearly identical performance for an array of task to standard Self- Attention yet with faster inference time and linear memory usage. However, beyond the memory complexity challenge of standard Self-Attention, there is also room for improvement in regard to how positional information can be embedded within the input.

[SUV18] proposes the use of relative positional encoding which is able to know the distance between elements of the input sequence instead of absolute positional encoding (APE which is what is proposed in [Vas+17]. By using relative positional encoding (RPE, [SUV18] further improvement was able to be achieved on an array of NLP task using RPE compared to using APE. This prompts the use of RPE instead of APE in order to achieve further performance gains however, due to it's formulation of embedding relative position into the query and key values of the Self-Attention, the issue of $O(N^2)$ complexity rises again. Fortunately, in the recent work of [Liu+21], a new proposed method called Stochastic Positional Encoding (SPE) which allows for RPE that is compatible with the proposed Linear Transformer from [Kat+20] allowing for RPE and Self Attention yet maintaining the memory complexity of $O(N)$.

This is possible by how SPE was formulated to have equivalency between positional encoding and cross-covariance structures of correlated Gaussian processes. In other words, [Liu+21] show that an attention kernel $P_d(m, n)$ can be formulated as a covariance matrix such that for all queries and key inputs, refered to as $(m, n)$, $P_d(m, n) =$

$\mathbb{E}[\bar{Q}_d(m)\bar{K}_d(n)]$ where $\bar{Q}_d(m)$ and $\bar{K}_d(n)$ are two real and zero-mean random variables. When this holds for the random variables, $\bar{Q}_d(m)$ and $\bar{K}_d(n)$, the random encoding postions for m queries and n keys yield the attention kernel on average. To enforce this behavior in the random variables, we use in this work the SineSPE conditions derived in [Liu+21] that proposes how to construct $\bar{Q}_d(m)$ and $\bar{K}_d(n)$ so that $P_d(m,n) = \mathbb{E}[\bar{Q}_d(m)\bar{K}_d(n)]$ holds. The formulation of SPE and SineSPE are shown below.

$$\hat{Q} \leftarrow \sum_{d=1}^{D} diag(q_{:,d})\bar{Q}_d/\sqrt[4]{DR}$$
$$\hat{K} \leftarrow \sum_{d=1}^{D} diag(k_{:,d})\bar{K}_d/\sqrt[4]{DR}$$

Equation set 1: SPE

$$\begin{aligned} \bar{Q} &\leftarrow \Omega(M, f_d, \theta_d) diag(\ddot{\lambda}_d)^2 Z_d/\sqrt{ZK} \\ \bar{K} &\leftarrow \Omega(M, f_d, 0) diag(\ddot{\lambda}_d)^2 Z_d/\sqrt{ZK} \\ Z_d \sim ZK \times R &\text{ are i.i.d draws from a standard Gaussian} \\ P_d &= \Omega(M, f_d, \theta_d) diag(\ddot{\lambda}_d)^2 \Omega(N, f_d, 0)^T \\ P_d &\approx \mathbb{E}[\bar{Q}(m)\bar{K}(n)] \text{ for large R on average} \end{aligned}$$

$$[\Omega(I,a,b)]_{nl} = \begin{cases} cos(2\pi a_k n + b_k) & \text{if } l = 2k \\ sin(2\pi a_k n + b_k) & \text{if } l = 2k+1 \end{cases}$$

Equation set 2: SineSPE

## 8 Methodology: Rosie's Hearing Aid

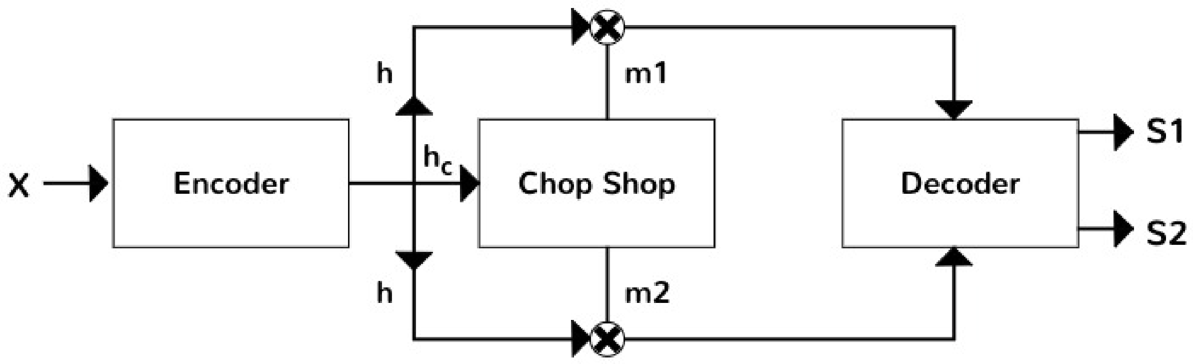

Figure 1: RHA for two mixed sources

### 8.1 Encoder

As in [Sub+21], the encoder takes in as input a time-domain mixture-signal $x \in \mathbb{R}^T$, which consist of multiple sources that are desired to be separated. By using a single convolution layer, the model is able to learn a STFT-like representation $h \in \mathbb{R}^{F \times T'}$ [Sub+21].

$$h = ReLU(conv1d(x))$$

After the initial 1d convolution, a linear dense layer of shape $F \times F$ is applied, after which a layer normalization is applied.

$$h' = LayerNorm(Dense(h))$$

To finish off the encoder block, a chunking operation is applied Following the works of [Sub+21], the chunking operation creates overlapping chunks of size C by chopping up $h'$ along the time-axis with using an overlapping factor of 50%. The output of the chunking operation is $h_c \in \mathbb{R}^{F \times C \times N_c}$ where $C$ is the length of each chunk and $N_c$ is the total number of chunks produced.

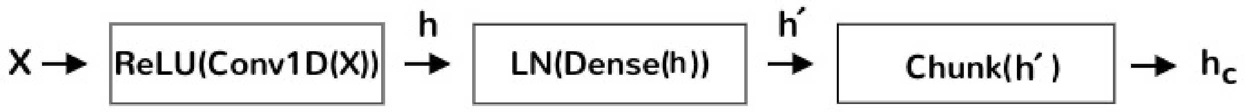

Figure 2: Encoder block

## 8.2 Chop Shop

## 8.3 Decoder

## 8.4 Loss Function

Si-SNR $= 10\log_{10}\frac{||x_{target}||^2}{||e_{noise}||^2}$, where $x_{target} = \frac{<\hat{x},x>x}{||x||^2}$, $e_{noise} = \hat{x} - x_{target}$, with pit [Kol+17]

$$SDR = 10\log_{10}\frac{||x_{target}||^2}{||e_{inter} + e_{noise} + e_{artif}||^2}$$

$$Si - SNRi = sisnr(sep, sep - est) - sisnr(sep, mix), \text{ show average over n-sources}$$

## 9 Experiments Setup

## 10 Results

### 10.1 Results on Fuss

| Name | Model | Stride | # Param | SI-SNRi | SDRi |
|---|---|---|---|---|---|
| fa | Linear Sepformer | 8 | | | |
| fb | Linear Sepformer | 4 | | | |
| ... | | | | | |
| ... | | | | | |
| | Sepformer | 8 | | | |
| | Sepformer | 4 | | | |

### 10.2 Results on LibriMix

| Name | Model | Stride | # Param | SI-SNRi | SDRi |
|---|---|---|---|---|---|
| la | Linear Sepformer | | | | |
| lb | Linear Sepformer | | | | |
| ... | | | | | |
| ... | | | | | |
| | Sepformer | | | | |
| | Sepformer | | | | |

### 10.3 Ablation Study

| Name | DM | SineSPE | N Blocks | N Tran | # Heads | # Param | SI-SNRi | SDRi |
|---|---|---|---|---|---|---|---|---|
| aa | No | Yes | 2 | 8 | 8 | | | |
| ab | Yes | Yes | 2 | 8 | 8 | | | |
| ac | Yes | No | 2 | 8 | 8 | | | |
| | Yes | Yes | 4 | 8 | 8 | | | |
| | Yes | Yes | 4 | 16 | 8 | | | |
| | Yes | Yes | 4 | 8 | 16 | | | |
| ... | | | | | | | | |
| ... | | | | | | | | |

### 10.4 Inference Time Comparison

### 10.5 Memory Usage Profiling

## 11 Future Work

## A Stochastic Positional Encoding

## References

[Kol+17] Morten Kolbæk et al. "Multitalker speech separation with utterance-level permutation invariant training of deep recurrent neural networks". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25.10 (2017), pp. 1901–1913.

[Vas+17] Ashish Vaswani et al. *Attention Is All You Need.* 2017. arXiv: `1706.03762 [cs.CL]`.

[SUV18] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. *Self-Attention with Relative Position Representations.* 2018. arXiv: `1803.02155 [cs.CL]`.

[Rae+19] Jack W. Rae et al. *Compressive Transformers for Long-Range Sequence Modelling.* 2019. arXiv: `1911.05507 [cs.LG]`.

[SLL19] David R. So, Chen Liang, and Quoc V. Le. *The Evolved Transformer.* 2019. arXiv: `1901.11117 [cs.LG]`.

[Suk+19] Sainbayar Sukhbaatar et al. *Adaptive Attention Span in Transformers.* 2019. arXiv: `1905.07799 [cs.LG]`.

[Dai+20] Zihang Dai et al. *Funnel-Transformer: Filtering out Sequential Redundancy for Efficient Language Processing.* 2020. arXiv: `2006.03236 [cs.LG]`.

[Kat+20] Angelos Katharopoulos et al. *Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention.* 2020. arXiv: `2006.16236 [cs.LG]`.

[KKL20] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. *Reformer: The Efficient Transformer.* 2020. arXiv: `2001.04451 [cs.LG]`.

[Cho+21] Krzysztof Choromanski et al. *Rethinking Attention with Performers.* 2021. arXiv: `2009.14794 [cs.LG]`.

[Jae+21] Andrew Jaegle et al. *Perceiver: General Perception with Iterative Attention.* 2021. arXiv: `2103.03206 [cs.CV]`.

[Liu+21] Antoine Liutkus et al. *Relative Positional Encoding for Transformers with Linear Complexity.* 2021. arXiv: `2105.08399 [cs.LG]`.

[Sub+21] Cem Subakan et al. *Attention is All You Need in Speech Separation.* 2021. arXiv: `2010.13154 [eess.AS]`.