Music Source Separation Using MMDenseNet with Transformer Encoder Bottleneck

Claire Jin
Artificial Intelligence
Carnegie Mellon University
Pittsburgh, PA 15213
claireji@andrew.cmu.edu

Mustafa Yilmaz
Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
mustafay@andrew.cmu.edu

Alan Zhu
Statistics and Data Science
Carnegie Mellon University
Pittsburgh, PA 15213
yixuanz2@andrew.cmu.edu

1 Introduction

There are many situations in which we may want to separate out a set of source signals from some input mixed signal without extra information or specific instructions for the mixing process. For example, to listen to a single conversation happening in a busy room, the human brain can easily separate the received audio signal and find the voices of the people in that conversation. This is an example of audio source separation, specifically speech separation, and applications include extracting the audio of a speaker in a loud area for captioning to help deaf people or isolating the speaker's voice in a phone call from background noise.

During music production and in live recordings of musical performances, the individual audio streams (called stems) of vocals and each individual instrument are mixed together to create a single audio track. The task of music source separation (MSS) is to recover the original individual stems. This problem differs from other audio source separation problems, like the separation of a voice from background noise, because in music there are more categories of audio sources with a wider variety of characteristics and a greater relationship with each other. Unlike distinguishing between human voice and noise, the musical stems are each equally important with a wide variety of tones and timbres [10], and they change in coordination with each other to produce a cohesive piece. Potential applications of MSS include separating out instruments from vocals for karaoke tracks [1], separating bass from soprano lines so users of music streaming platforms like Spotify can adjust the strength of each component to their personal liking, and removing background noise from musical audio for people who are hard of hearing or to improve the recording quality of live performances.

The audio input for MSS can either be represented as a waveform – a digitized audio signal similar to the physical sound wave – or as a spectrogram – a 2D time-frequency (TF) matrix [9]. We use the spectrogram input, as most existing approaches take spectrogram inputs. To perform MSS on spectrograms, we learn a mask for each target source being separated that conceals magnitude values in the spectrogram at each time step, assigning some portion of the mix's energy at that time to the corresponding source.

2 Background

Since 2017, the primary base model for spectrogram-based MSS neural networks has been a model which adapts the U-Net architecture [14] for vocal separation [6]. At a high level, this model

10617 Final Report (2022).

compresses the input into smaller, deeper representations through a series of 2D convolutional layers before deconvolving the result back to the original dimensions. Our work expands upon this general model structure by enhancing the compression and decompression processes, as well as the work done in the low dimensional space after compression, which we will refer to as the "bottleneck" of the model.

In our midway report, we implemented the MDenseNet model [16], a modification of DenseNet [5] with an additional decoder architecture, and performed vocal separation. The MDenseNet model comprises 8 dense blocks, each having 4 layers that each perform batch normalization, ReLU, and a 3x3 convolution. In the encoder between adjacent dense blocks is an average pooling downsample layer. In the decoder between adjacent dense blocks is an upsample layer performing batch normalization followed by a transposed convolution.

For the midterm report, we trained the model on 80 songs and evaluated using 14 different songs in the MUSDB18 dataset [13], sampling 5-seconds per song for both training and validation. Validation performance started to plateau after around 10 epochs. We observed a best training loss of 3.01 and a best validation loss of 3.98, and the final model had a Source-to-Distortion Ratio (SDR) of 3.47dB from the validation ground truth.

Subjectively comparing the model-separated vocal tracks to the original vocal stems, the vocals could be clearly heard in the output tracks. However, the model output tracks tended to miss some bass frequencies in the original vocal stems. Additionally, the model was not able to cleanly remove some high-frequency sounds, such as hi-hats and piano.

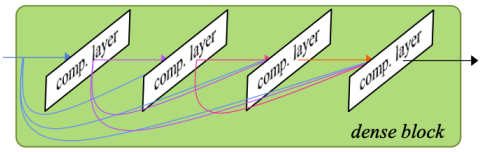
3 Related Work

3.1 Model Components

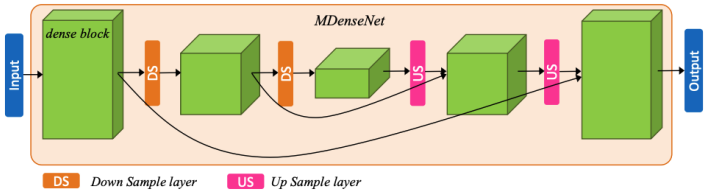
Though the iterative connections in the dense blocks (Figure 1a) of MDenseNet (Figure 1b) are an appropriate architecture for capturing the temporal and frequency relationships in music, they result in a very memory-expensive model since the number of inter-layer connections grows quadratically with the depth of the model [16]. Additionally, as we saw with the missing bass frequencies, some frequency bands tend to be missed by the model. To solve these problems, MMDenseNet (Figure 1c) [16], which utilizes dense blocks with low resolutions in addition to high resolutions and specializes dense blocks to particular frequency bands, was proposed. This model gives some dense blocks a lower resolution input created by down-sampling outputs from the previous dense blocks, reducing the number of connections needed in these lower resolution blocks. The low resolution outputs of these blocks are then up-sampled and fed into subsequent high resolution dense blocks along with the high resolution output from the previous high resolution dense blocks. Additionally, a sequence of dense blocks and encoder-decoders, forming an MDenseNet, are dedicated to each frequency band, since patterns in the spectrogram tend to differ between frequency bands and often certain bands become neglected during learning [16].

Since music has strong contextual dependencies both over time and frequencies, several methods have been proposed to leverage attention based Transformers [18] for integrating these informations. One such method, MTFAttNet [2], uses temporal and frequency self-attention layers in combination with additional convolutions to construct a separator module used as the bottleneck of the model. Another approach, Sams-Net [7], uses a multi-headed attention module and a slicing module, which applies a separate attention module to each non-overlapping temporal slice of the input spectrogram, as the bottleneck between several convolution and deconvolution layers.

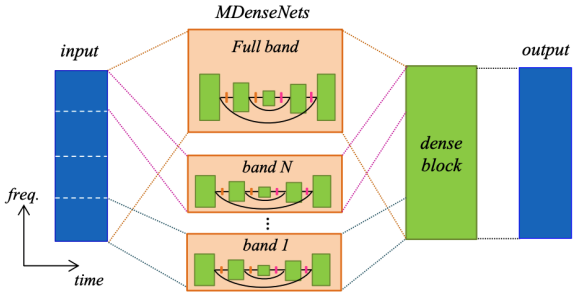
Another challenge of MSS is constructing a training objective that results in separations that are perceived as "good" by humans. A study of several commonly used loss functions benchmarked 10 different spectrogram-based loss functions when used for training the OpenUnmix model [15] and conducted controlled subjective evaluations of 3 of these losses [4]. They found the L2, LOGL2, scale-invariant SDR, and LOGL1 losses to perform similarly well and outperform all other spectrogram losses tested, without much benefit from combining these losses. These losses also outperformed the deep feature and adversarial losses tested.



(a) Dense Block



(b) MDenseNet



(c) MMDenseNet

Figure 1: MDenseNet and MMDenseNet architectures.

3.2 Data Augmentation

Data augmentation is a powerful tool for improving deep learning performance with limited data. SpecAugment is a popular technique for audio data augmentation [11]. The augmentation policy has three parts: random warping of the 2D spectrogram, randomly masking certain frequency channels, and randomly masking certain time steps. Though originally developed for speech recognition, these techniques have also been used for vocal separation with marginal, though statistically significant, improvements in performance [12].

4 Methods

4.1 Data

We used the MUSDB18 dataset [13], as this is a widely used and relatively large dataset in MSS, allowing us to easily compare our approach to others like MTFAttNet and Sams-Net [7, 2]. The dataset contains 150 full length music tracks (∼10h total duration) of different genres along with their isolated "drums", "bass", "vocals", and "others" stems. The dataset is curated with a training set of 100 songs, where 14 of these songs are designated for validation, and a test set of 50 songs.

For our data augmentation approach, we used only the frequency masking method described in SpecAugment [11] since frequency or pitch alterations were found to consistently improve performance in one study [12]. For each training audio mixture, we mask frequency channels $[f_0, f_0 + f)$, where f is uniformly randomly chosen from 0 to a parameter F, which we set equal to the number of frequency channels v, and f_0 is uniformly randomly chosen from $(0, v - f)$. To mask a frequency

channel, we set all values in the channel over all timesteps to the average value of that channel. We performed augmentation on the fly during training, creating one new datapoint from each original datapoint for each epoch.

4.2 Model

To establish a baseline, we trained MDenseNet models [16], largely following the original architecture. To improve upon this baseline, we trained MMDenseNet models [16]. Diverging from the original architecture, we split the input spectrogram into three bands: low, middle, and high. Each split of the spectrogram goes through its own individual MDenseNet architecture, before being concatenated in the frequency dimension. The full spectrogram also goes through a separate MDenseNet architecture, and then is stacked with the concatenated output from individual bands, before going through another dense block and 2D convolution.

We also propose a new architecture MMTDenseNet (Multi-scale Multi-band DenseNets with Transformer Encoder Bottleneck). In MMTDenseNet, we replace the dense block in the bottleneck with a Transformer Encoder [18] for each MDenseNet (i.e. for the individual bands and full spectrogram). Each Transformer Encoder contains two Transformer Encoder Layers, each using 4 attention heads on the frequency dimension and a feedforward dimension of 128. Detailed network architectures can be seen in Table 1, where k and L stand for the growth rate and number of layers in each dense block, respectively. Additionally, all convolution operations use kernel size 3x3, and the average pooling and transposed convolution operations use kernel size 2x2 and stride 2. For each convolution operation, we also use batch normalization and ReLU activation.

Layer		MDenseNet	MMDenseNet				MMTDenseNet			
			Low	Middle	High	Full	Low	Middle	High	Full
Band Split (frequency channels)		None	380	644	1025	None	380	644	1025	None
Conv (channels)		32	32	32	32	32	32	32	32	32
Encoder	Dense 1 (k, L) Downsample	12, 4 Pool	10, 3 Pool	10, 3 Pool	10, 3 Pool	10, 3 Pool	10, 3 Pool	10, 3 Pool	10, 3 Pool	10, 3 Pool
	Dense 2 (k, L) Downsample	12, 4 Pool	10, 3 Pool	10, 3 Pool	10, 3 Pool	10, 3 Pool	10, 3 Pool	10, 3 Pool	10, 3 Pool	10, 3 Pool
	Dense 3 (k, L) Downsample	12, 4 Pool	10, 3 Pool	10, 3 Pool	10, 3 Pool	10, 3 Pool	4, 3 Pool	4, 3 Pool	4, 3 Pool	4, 3 Pool
Bottleneck	Dense 4 (k, L) / Transformer Encoder	12, 4	10, 3	10, 3	10, 3	10, 3	Transformer Encoder	Transformer Encoder	Transformer Encoder	Transformer Encoder
Decoder	Upsample Concat. (skip) Dense 5 (k, L)	t.conv Dense 3 12, 4	t.conv Low Dense 3 10, 3	t.conv Middle Dense 3 10, 3	t.conv High Dense 3 10, 3	t.conv Full Dense 3 10, 3	t.conv Low Dense 3 10, 3	t.conv Middle Dense 3 10, 3	t.conv High Dense 3 10, 3	t.conv Full Dense 3 10, 3
	Upsample Concat. (skip) Dense 6 (k, L)	t.conv Dense 2 12, 4	t.conv Low Dense 2 10, 3	t.conv Middle Dense 2 10, 3	t.conv High Dense 2 10, 3	t.conv Full Dense 2 10, 3	t.conv Low Dense 2 10, 3	t.conv Middle Dense 2 10, 3	t.conv High Dense 2 10, 3	t.conv Full Dense 2 10, 3
	Upsample Concat. (skip) Dense 7 (k, L)	t.conv Dense 1 12, 4	t.conv Low Dense 1 10, 3	t.conv Middle Dense 1 10, 3	t.conv High Dense 1 10, 3	t.conv Full Dense 1 10, 3	t.conv Low Dense 1 10, 3	t.conv Middle Dense 1 10, 3	t.conv High Dense 1 10, 3	t.conv Full Dense 1 10, 3
	Concat. (axis)	None	Frequency			None	Frequency			None
	Concat. (axis)	None	Channel				Channel			
Output	Dense 8 (k, L)	4, 2	4, 2	4, 2	4, 2	4, 2	4, 2	4, 2	4, 2	4, 2
	Conv (channels)	2	2	2	2	2	2	2	2	2

Table 1: Network architecture details of our MDenseNet, MMDenseNet, MMTDenseNet models.

4.3 Loss Functions

We trained our models using standard regression loss functions, namely L1 and L2 loss. The loss is computed between the spectrograms of the target and the model output. This involves performing a short-time Fourier transform on the time-domain signals. We apply the loss functions to only the magnitude spectrograms.

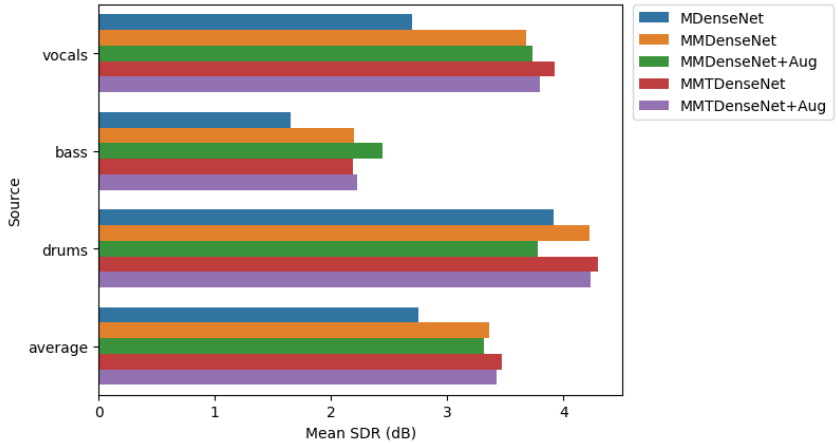
4.4 Training and Evaluation

We trained separate networks for separating each source (vocals, bass, drums), all following a standard procedure. We trained for 150 epochs using the designated training portion of the MUSDB18 dataset, with batch size 8 and an Adam optimizer with learning rate 0.001. For each batch we randomly sampled 5 contiguous seconds from one stereo track and applied short-time Fourier transform (STFT) with fast Fourier transform (FFT) size 4096, hop length 1024, and Hanning window. We then directly applied frequency masking on the output if using data augmentation.

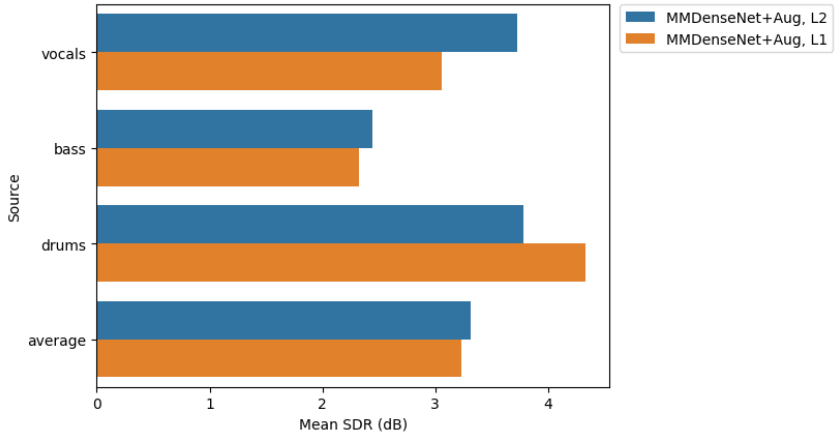
We input the magnitude spectrogram (real outputs of STFT) to our network and then calculate the loss between the output and the magnitude spectrogram of the target source track. For validation, we used the full tracks in the validation split of MUSDB18. We compute loss using the magnitude spectrogram without augmentation. We run validation every 10 epochs, and choose the model with lowest validation loss as our final model.

For testing, we used the full tracks in the testing split of MUSDB18, using the full procedure of source separation as follows. For each track, we take the mixture waveform, apply STFT, run the network on the magnitude spectrogram, combine output with original phase (complex outputs of STFT) of the mixture, and apply inverse short-time Fourier transform (iSTFT) to get the waveform output of the source-separated track.

5 Results



(a) Comparisons of model performances in SDR across sources.



(b) Comparisons of model performances using L1 and L2 loss.

Figure 2: Comparisons of model performances.

We assessed the performance of our MSS models using the average SDR of the 50 test tracks in MUSDB18. We used SDR, as this is the standard evaluation metric in MSS. On average, the MMTDenseNet without augmentation performed the best.

Our baseline model from the midway report was MDenseNet without frequency masking and trained using L2 loss. The next model we tried was MMDenseNet, again without frequency masking and trained using L2 loss. This resulted in substantial improvements in SDR across all three sources as

Model	Data Augmentation	Loss	Vocal SDR	Bass SDR	Drums SDR
MDensenet	None	L2-Freq	2.697	1.650	3.919
MMDensenet	None	L2-Freq	3.675	2.194	4.222
MMDensenet	Frequency Mask	L2-Freq	3.730	2.445	3.778
MMTDensenet	None	L2-Freq	3.923	2.184	4.296
MMTDensenet	Frequency Mask	L2-Freq	3.801	2.227	4.234
MMDensenet	Frequency Mask	L1-Freq	3.057	2.320	4.328

Table 2: Summary of models and their SDR performances for different source modalities

we expected since the capacity of MMDenseNet is higher and the original MMDenseNet paper found similar results [16].

We were able to improve the performance of MMDenseNet modestly using frequency masking data augmentation for vocals and bass separation, but the performance suffered when separating drums. We expected the performance to either increase or stay the same since previous work analyzing the effects of data augmentation in MSS found only marginal increases in performance when using frequency-based data augmentation [12]. Thus, the vocals and bass results are as we expected, but the drums results are surprising, especially since the MMDenseNet with frequency masking performed worse than MDenseNet for drums.

Without frequency masking, using a Transformer Encoder bottleneck (MMTDenseNet) improved SDR greatly for vocals, modestly for drums, and resulted in no improvement for bass. This result is in line with our expectations, since adding a Transformer bottleneck should improve integration of the contextual information throughout the audio mixture. Adding frequency masking to MMTDenseNet did not prove to be useful, as performance decreased for all sources except bass.

We also assessed the effect of the loss function by training MMDenseNet with frequency masking using L1 loss. The resulting model significantly outperformed the one trained using L2 loss for drums, but did worse for both vocals and bass.

We were unable to compare our model performance directly to the original MDenseNet and MMDenseNet models, as they were trained and tested on different datasets. However, our source separation models perform worse than recent MSS models trained on MUSDB18, such as OpenUnmix [15] and Demucs [3], and only outperform some models such as Wave-U-Net [8].

6 Discussion and Analysis

We found that through adding multiple bands to MDenseNet in MMDenseNet, performance improved drastically. This enforces existing beliefs that different patterns are important in different frequency bands. Our work with MMTDenseNet also shows that incorporating attention-based architectures, such as the Transformer Encoder, increases the performance of MSS models. This is likely due to the importance of relationships between various stems, each of which tend to stay in specific frequency bands, in musical pieces. Our Transformer Encoder allows for better contextualization of frequencies among the other frequencies existing at that time step.

Due to limits in computing resources, we had to change the growth rate of the dense block in the third downsampling block in our MMTDenseNet model from 10 to 4, which gives the transformer encoder bottleneck less information than the dense block bottleneck in MMDenseNet, resulting in a less fair comparison. However, given enough computing budget, we expect using the original growth rate would lead to better performance for the MMTDenseNet model.

We also found that data augmentation does not always improve performance and that most observed improvements are relatively small. However, it's possible that adding more varieties of augmentation, such as time masking or channel swapping, would result in more diverse data augmentations and change this result. Additionally, we set the parameter F to be the total number of frequency channels, but lowering this parameter may yield better data augmentation results, so we would only mask a small number of frequency channels during augmentation.

We observed that our models usually perform well for one or two sources, but not for all sources. Specifically, the performance for separating drums tended to be the most different from the results for

vocals and bass. This suggests that the underlying structure of drum stems may be different from other instruments. Some potentially important differences could be that the drum parts are highly rhythmic and can play in a wider range of frequencies compared to other instruments. For example, hi-hats have a high pitch while bass drums have a very low pitch.

With this in mind, we believe doing further hyperparameter tuning and individualized neural architecture search for each source would result in better training performance and higher test SDR for each source. Furthermore, using other optimization techniques, such as learning rate decay, may result in better convergence. In terms of loss functions, in addition to using L1 or L2 losses in the frequency domain, we could explore losses in the time domain (waveform loss) and combine multiple loss functions. To further improve the quality of source-separated audio outputs, we can also apply multi-channel Wiener filters [17] to the waveform before converting it to a spectrogram input for the model.

References

[1] Kevin Brown. *Karaoke Idols: Popular Music and the Performance of Identity*. London: Intellect Books, 2015.

[2] Lianwu Chen et al. *Multi-scale temporal-frequency attention for music source separation*. 2022. DOI: 10.48550/ARXIV.2209.00805. URL: https://arxiv.org/abs/2209.00805.

[3] Alexandre Défossez. *Hybrid Spectrogram and Waveform Source Separation*. 2021. DOI: 10.48550/ARXIV.2111.03600. URL: https://arxiv.org/abs/2111.03600.

[4] Enric Gusó et al. *On loss functions and evaluation metrics for music source separation*. 2022. DOI: 10.48550/ARXIV.2202.07968. URL: https://arxiv.org/abs/2202.07968.

[5] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. "Densely Connected Convolutional Networks". In: *CoRR* abs/1608.06993 (2016). arXiv: 1608.06993. URL: http://arxiv.org/abs/1608.06993.

[6] A. Jansson et al. "Singing voice separation with deep U-Net convolutional networks". Oct. 2017. URL: https://openaccess.city.ac.uk/id/eprint/19289/.

[7] Tingle Li et al. "Sams-Net: A Sliced Attention-based Neural Network for Music Source Separation". In: *2021 12th International Symposium on Chinese Spoken Language Processing (ISCSLP)*. 2021, pp. 1–5. DOI: 10.1109/ISCSLP49672.2021.9362081.

[8] Francesc Lluís, Jordi Pons, and Xavier Serra. *End-to-end music source separation: is it possible in the waveform domain?* 2018. DOI: 10.48550/ARXIV.1810.12187. URL: https://arxiv.org/abs/1810.12187.

[9] Ethan Manilow, Prem Seetharman, and Justin Salamon. *Open Source Tools & Data for Music Source Separation*. https://source-separation.github.io/tutorial, 2020. URL: https://source-separation.github.io/tutorial.

[10] Nicola Orio. "Music Retrieval: A Tutorial and Review". In: *Found. Trends Inf. Retr.* 1.1 (Jan. 2006), pp. 1–96. ISSN: 1554-0669. DOI: 10.1561/1500000002. URL: https://doi.org/10.1561/1500000002.

[11] Daniel S. Park et al. "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition". In: *Interspeech 2019*. ISCA, Sept. 2019. DOI: 10.21437/interspeech.2019-2680. URL: https://doi.org/10.21437%5C%2Finterspeech.2019-2680.

[12] Laure Pretet et al. "Singing Voice Separation: A Study on Training Data". In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, May 2019. DOI: 10.1109/icassp.2019.8683555. URL: https://doi.org/10.1109%5C%2Ficassp.2019.8683555.

[13] Zafar Rafii et al. *The MUSDB18 corpus for music separation*. Dec. 2017. DOI: 10.5281/zenodo.1117372. URL: https://doi.org/10.5281/zenodo.1117372.

[14] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *CoRR* abs/1505.04597 (2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597.

[15] Fabian-Robert Stöter et al. "Open-Unmix - A Reference Implementation for Music Source Separation". In: *Journal of Open Source Software* 4.41 (2019), p. 1667. DOI: 10.21105/joss.01667. URL: https://doi.org/10.21105/joss.01667.

[16] Naoya Takahashi and Yuki Mitsufuji. "Multi-scale Multi-band DenseNets for Audio Source Separation". In: *CoRR* abs/1706.09588 (2017). arXiv: 1706.09588. URL: http://arxiv.org/abs/1706.09588.

[17] Stefan Uhlich et al. "Improving music source separation based on deep neural networks through data augmentation and network blending". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017, pp. 261–265. DOI: 10.1109/ICASSP.2017.7952158.

[18] Ashish Vaswani et al. *Attention Is All You Need*. 2017. DOI: 10.48550/ARXIV.1706.03762. URL: https://arxiv.org/abs/1706.03762.