

Understanding Audio Features via Trainable Basis Functions

Kwan Yee Heung¹, Kin Wai Cheuk^{1,2}, Dorien Herremans¹

¹Singapore University of Technology and Design

²Agency for Science, Technology and Research

{kwanyee_heung, dorien_herremans}@sutd.edu.sg, kinwai_cheuk@mymail.sutd.edu.sg

Abstract

In this paper we explore the possibility of maximizing the information represented in spectrograms by making the spectrogram basis functions trainable. We experiment with two different tasks, namely keyword spotting (KWS) and automatic speech recognition (ASR). For most neural network models, the architecture and hyperparameters are typically fine-tuned and optimized in experiments. Input features, however, are often treated as fixed. In the case of audio, signals can be mainly expressed in two main ways: raw waveforms (time-domain) or spectrograms (time-frequency-domain). In addition, different spectrogram types are often used and tailored to fit different applications. In our experiments, we allow for this tailoring directly as part of the network.

Our experimental results show that using trainable basis functions can boost the accuracy of Keyword Spotting (KWS) by 14.2 percentage points, and lower the Phone Error Rate (PER) by 9.5 percentage points. Although models using trainable basis functions become less effective as the model complexity increases, the trained filter shapes could still provide us with insights on which frequency bins are important for that specific task. From our experiments, we can conclude that trainable basis functions are a useful tool to boost the performance when the model complexity is limited.

Index Terms: speech recognition, keyword spotting, trainable front-ends, feature extraction

1. Introduction

Feature engineering has been a crucial part of developing a good machine learning model. A range of different audio front-end toolkits have been developed for this purpose in the field of audio [1, 2, 3, 4, 5]. From choosing which features to include [6, 7] to allowing the model to learn its own features [8, 9, 10], researchers have been trying to extract as much useful information from audio clips as possible. Choosing the most important features from audio signals is more difficult than from images, since audio features can be expressed in either time-domain or time-frequency-domain. Even more so, there exists more than one time-frequency-domain feature representation, for instance, short-time Fourier transform (STFT), constant-Q transform (CQT) [11], and Mel-frequency cepstral coefficients (MFCC) [12]. Due to the effort required to select important features, some researchers simply use the same spectrogram configuration as in related literature [13, 14, 15, 16], hoping that a deep enough model could extract all the information it needs for the task regardless of what audio features are fed to the model.

nnAudio [17] is a trainable front-end tool which allows the optimization of raw waveforms (time-domain) to spectrograms (time-frequency-domain) conversion. This is achieved by implementing the STFT and Mel basis functions as a first layer of the neural network (front-end layer), which allows models to

back-propagate the gradient to the front-end. And hence, a custom spectrogram can be ‘calculated’ that suits the task better.

Similar to nnAudio, FastAudio [18], is also a trainable front-end tool but with shape constraint applied to Mel bases. In their spoofing detection model, the trainable Mel bases results in a higher accuracy than using conventional spectrograms. Yet, spoofing detection is a binary classification [19, 20, 21], and it is still unclear if the same applies to other tasks. In this paper, we extend the idea of trainable basis functions to more tasks such as keyword spotting (KWS) and automatic speech recognition (ASR). More specifically, we want to demonstrate the following two points: 1) Under what circumstances are trainable basis functions useful? 2) What basis functions are learned after training and why are they better than the original one?

2. Setup

To address the research questions above, a number of experiments are conducted in which we compare the performance of trainable short-time Fourier transform (STFT) g_{STFT} and Mel basis functions g_{Mel} provided by FastAudio and nnAudio on two tasks: KWS and ASR. We use a broadcasting-residual network (BC-ResNet) [22] as well as a Simple model (constructed with a linear layer) for these two tasks.

nnAudio can produce the same STFT outputs as other major audio processing libraries such as librosa [3] and torchaudio [23], however, it allows g_{STFT} to be trained together with the model, the effect of which we aim to study.

2.1. Trainable basis functions

For both nnAudio [17] and FastAudio [18], 40 Mel bases ranging from 0Hz to 8,000Hz are used. We denote the audio front-end as g , to which we apply both g_{STFT} as well as g_{Mel} on the raw waveform $X_t \in [-1, 1]^{T_t}$, resulting in a spectrogram $X_f \in [0, +\infty)^{T_f \times 40}$:

$$\begin{cases} g = g_{\text{Mel}} \circ g_{\text{STFT}} \\ X_f = g(X_t) \end{cases}, \quad (1)$$

where T_t and T_f are the lengths of the sample in the time- and frequency-domain respectively.

Both nnAudio and FastAudio initialise the g_{Mel} as triangular shapes [24, 25, 26, 27]. One major difference between nnAudio and FastAudio is that all Mel bases g_{Mel} in FastAudio have the same amplitude, and the bases remain triangular shape in both the initial and training state. In nnAudio on the other hand, the amplitude of the initialised g_{Mel} decays as the frequency goes up. During the training state, nnAudio allows g_{Mel} to be updated to any shape and with any amplitude ranging from 0 to 1. In our experiments, we explore four different training settings:

A Both g_{Mel} and g_{STFT} are non-trainable.

B g_{Mel} is trainable while g_{STFT} is fixed.

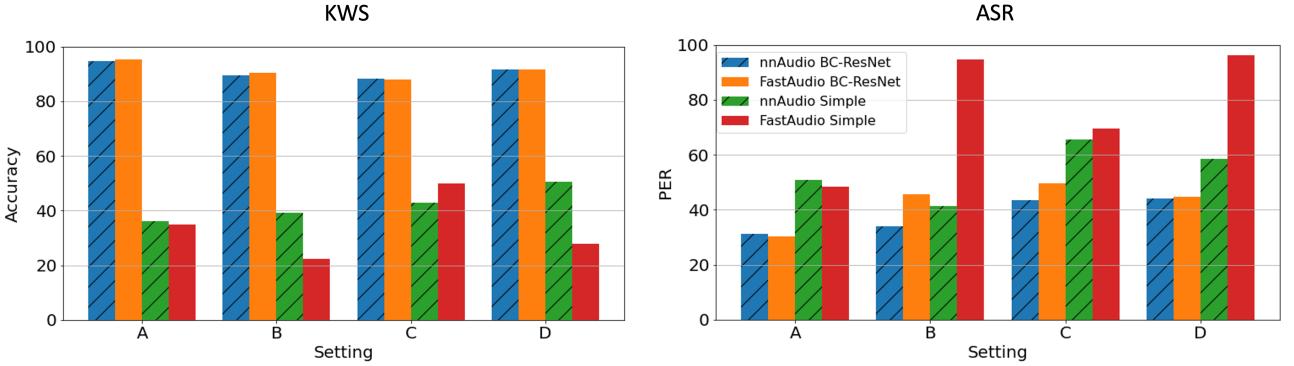


Figure 1: Model performances under four different settings (A-D defined in Section 2) on both KWS and ASR.

Table 1: Keyword spotting accuracy when different number of Mel bases are used under four different trainable settings A,B,C,D. (defined in Section 2)

# Mel	nnAudio					FastAudio				
	A	B	C	D	max. imp.	A	B	C	D	max. imp.
10	23.9%	37.7%	39.6%	34.5%	15.7 ppt.	35.0%	8.8%	36.3%	17.2%	1.3 ppt.
20	27.9%	37.4%	32.7%	38.6%	10.7 ppt.	31.8%	19.1%	46.2%	24.6%	14.4 ppt.
30	26.4%	39.7%	43.6%	40.4%	14.0 ppt.	32.2%	17.7%	50.0%	23.7%	17.8 ppt.
40	36.2%	39.3%	42.8%	50.4%	14.2 ppt.	35.0%	22.3%	49.8%	27.8%	14.8 ppt.

C g_{Mel} is fixed while g_{STFT} is trainable.

D Both g_{Mel} and g_{STFT} are trainable.

2.2. Preprocessing and model architecture

The same audio preprocessing pipeline is used throughout our experiments. All of the audio clips are downsampled into 16kHz. All experiments share the same STFT front-end g_{STFT} provided by nnAudio with a Hann window size of 480 samples, and a hop size of 160 samples. For g_{Mel} , 40 Mel bases are used.

For each setting, we consider two model architectures, BC-ResNet [22] and Simple, as the classifier which is denoted by f . Hence, the output of the classifier $Y \in [0, 1]^C$ can be obtained via:

$$Y = (f \circ g)(X_t) \\ = f(X_f), \quad (2)$$

where C is the number of classes depending on the task defined in Section 3.1 and Section 3.2.

3. Experiments

3.1. Keyword Spotting

We use the Google speech commands dataset v2 [28] to examine the effects of trainable basis functions. There are total 12 (C=12) different classes in this KWS task, and the dataset has total 35 single wordings. Following existing literature [22, 28, 29], 10 out of 35 words are chosen ('down', 'go', 'left', 'no', 'off', 'on', 'right', 'stop', 'up', 'yes'). The remaining 25 words are grouped as class 'unknown'. A class 'silence' is created from background noise as in existing literature [22, 28, 29]. Due to the class imbalance between the 'silence' and 'unknown' class, we re-balance the training set by adjusting the sampling weight for each class during training [22, 28, 29]. Softmax is applied to the model output Y and

cross entropy minimized using Adam optimizer with a constant learning rate of 1×10^{-3} . All models are trained for 200 epochs with a batch size of 100 as it yields the best accuracy.

Figure 1 shows the speech command prediction accuracy using different models and settings. When a linear layer (Simple) is used for the prediction (green bars), using a trainable g_{STFT} and g_{Mel} with nnAudio results in a higher accuracy. When both g_{STFT} and g_{Mel} are trainable (setting D), the accuracy is 14.2 percentage point higher than with fixed g_{STFT} and g_{Mel} (setting A). Interestingly, the shape constraints imposed by FastAudio (red bars in Figure 1) during training did not help in keyword spotting. On the contrary, shape constraints harmed the model performance. We also tried training the FastAudio front-end with different learning rates (from 1×10^{-5} to 1×10^{-3}) but none of it yields a result as good as nnAudio's. Therefore, the shape constraints might not work well for all speech-related tasks in general.

When BC-ResNet is used, the keyword spotting accuracy reaches over 90%, but the trainable basis functions (settings B-D) become less effective. We believe that when the model is deep enough, it is able to learn how to extract useful information out of the spectrograms, even if they are less tuned to the task. At the same time, a complex model is also very sensitive to the change of the input X_f , and therefore trainable kernels together with a complex model might potentially confuse the model instead of helping. When a shallow model is used, it does not have enough power to extract useful information out of X_f , hence trainable basis functions produce a better X_f so that a shallow model can achieve a better prediction accuracy. We conducted an ablation study to understand what happens when we reduce the number of Mel bases g_{Mel} in the spectrograms (see Table 1). In all cases, trainable basis functions (either setting B, C, or D) yield a better result than non-trainable basis functions (setting A). When nnAudio is used as the trainable g ,

it improves the accuracy by at least 10 percentage points (ppt.). Setting C and D on average yield a better model performance. It shows that trainable STFT kernels are able to extract more information out of the raw waveform X_t .

The shape constraint imposed by FastAudio does not improve the accuracy for this task. Whenever the g_{Mel} are updated with the shape constraint (setting B and D), the performance becomes worse than when keeping the basis functions non-trainable. Nonetheless, it is interesting to see that with trainable g_{STFT} (setting C) the model performance is relatively unchanged when the number of Mel bases in g_{Mel} is reduced from 40 to 30.

3.2. Automatic Speech Recognition

The TIMIT dataset [30] is used to study the effect of trainable basis functions on ASR. TIMIT contains 6,300 sentences from 630 speakers. We mainly focus on phoneme recognition by using the phoneme labels provided in the dataset. There are 61 distinct phoneme labels with 1 separator class, $C = 62$ in this task. We measure the model performance using the phone error rate (PER). We use a batch size of 100 to train the model for 400 epochs. The connectionist temporal classification (CTC) loss [31] is minimized using an Adam optimizer with a constant learning rate of 1×10^{-3} . As ASR is a much more difficult task than KWS, using a linear layer alone does not perform well. Therefore we use a long short-term memory (LSTM) layer [32] together with either a modified BC-ResNet [22] or a linear layer as the classifier. To preserve the time dimension, we modify the BC-ResNet by removing the average pooling along the time dimension such that the output of BC-ResNet has the same number of timesteps T_f as the input spectrogram X_f .

Similar to KWS, we only observe improvements when the model is a relatively simple (LSTM + linear layer). When BC-ResNet is used instead of a linear layer, using trainable g_{Mel} (setting B) worsens the PER from 31.2 to 33.9. Unlike KWS, trainable g_{Mel} (setting B) are more effective than trainable g_{STFT} (setting C). When nnAudio Simple is used, the former setting improves the PER from 51.0 to 41.4, and the latter worsen the PER to 65.58. We believe that due to the fact that ASR (one input, many predictions) is more complicated than KSW (one input, one prediction).

3.3. Trained Mel

To understand what spectral features has been learned, we visualize the changes in g_{Mel} after training on the speech commands dataset in Figure 2(a)-(b). The following discussion is based on Simple as the classifier with trainable g_{Mel} and fixed g_{STFT} .

Before training, the Mel bins become wider and weaker as the bin number goes up, and they attend to the STFT bins in a log relationship. After training, each Mel bin focuses less on the STFT bins that they originally attended to. Instead, they attend to more STFT bins than before. Figure 2(c) shows the cumulative importance of different STFT bins before and after training. This is calculated by summing all bases in g_{Mel} and then normalizing by the largest value. Before training, the g_{Mel} extract information mostly from STFT bins 0-30, and the importance gradually decays for the higher STFT bins. After training, the g_{Mel} still pay great attention to the first few STFT bins. But STFT bins 10-25 are not as important as before. Instead, STFT bin numbers above 125 are more important than before. To prove that this finding is not due to the initial states of the g_{Mel} leading, we randomly initialized the basis functions, and trained these randomly initialized basis functions end-to-

end together with the model. The result is shown as the green lines of Figure 2(c), which converges to the same pattern as the one initialized as triangular g_{Mel} (orange line). We observe a very similar pattern after training on the TIMIT dataset, but g_{Mel} tend to pay attention to different frequency bins.

In Table 2, we apply masks to the STFT output to study the effects on the prediction accuracy. Figure 4 shows the cumulative STFT bin importance that g_{Mel} pays attention to. When STFT bins are masked by 0, the g_{Mel} ignores those bins. In the case of KWS, the original accuracy is 42.8% without applying any mask to the STFT output. When masking bin numbers 25-49, the accuracy increases to 45.6%. Masking more frequency bins (bin numbers 25-74) makes the accuracy worse than before, which indicates that bin numbers 50-74 contain important information for this task. When bin numbers 216-240 are masked, the model reaches its best accuracy (49.45%). A similar pattern can be observed for the ASR task.

3.4. Trained STFT

Similar to Section 3.3, we found that trained g_{STFT} also pay more attention to lower frequencies than to higher frequencies.

Figure 3(a) shows the time-domain STFT filter corresponding to STFT bin number 25 (833 Hz). The original imaginary part of STFT kernel (blue line) is a pure tone containing only one frequency component. The discrete Fourier transform (DFT) in Figure 3(b) shows that this pure sine wave has only one frequency component (833 Hz). After training, we can see that the amplitude of the sine filter is lower than the original one. While the trained filter maintains the same sine wave shape as its backbone, it superimposes higher frequencies components which can be seen from the DFT analysis in Figure 3(b). Rather than the original peaks, smaller peaks emerge around it.

Figure 3(c) shows the the cumulative importance of different frequencies. Similar to g_{Mel} , all of the frequency components from each STFT filter are summed together and then normalized by the maximum value such that the cumulative importance is in the range of $[0, 1]$. The overall trend of the cumulative importance is very similar to Figure 2(c), where frequencies between 1000Hz and 2000Hz are more important than frequencies above 7000Hz. We observe a very similar pattern for ASR, and are therefore not repeating the same discussion here due to page limits.

3.5. Corrupted Spectrograms

Another interesting property of trainable g_{Mel} is that it excels when some frequency bins on the STFT output are missing. Here, we keep STFT fixed and train the g_{Mel} . Table 2 shows the model performance when we mask out different STFT frequency bins with the value 0. When STFT bins 216-240 are masked, a simple model with trainable g_{Mel} is able to achieve an accuracy of 49.5% for KWS, which is much higher than 42.8% (the result when all STFT frequency bins are available). However, the same mask applied when g_{Mel} is in a non-trainable state, causes the KWS accuracy to deteriorate to only 18.3%. A similar pattern can be observed for the ASR task. It is notable that trainable Mel together with missing STFT bins achieves a better result than when all STFT bins are intact.

In Figure 4, we try to understand what is happening when some STFT bins are masked out. When bins 25-49 are missing, the trained g_{Mel} pays more attention to bin 24 and 50 and ignore bins 25-49. When bins 216-240 are masked, the trained Mel bases pay attention to bins 200-215 more than before. We believe that the trainable g_{Mel} is able to compensate for the loss of

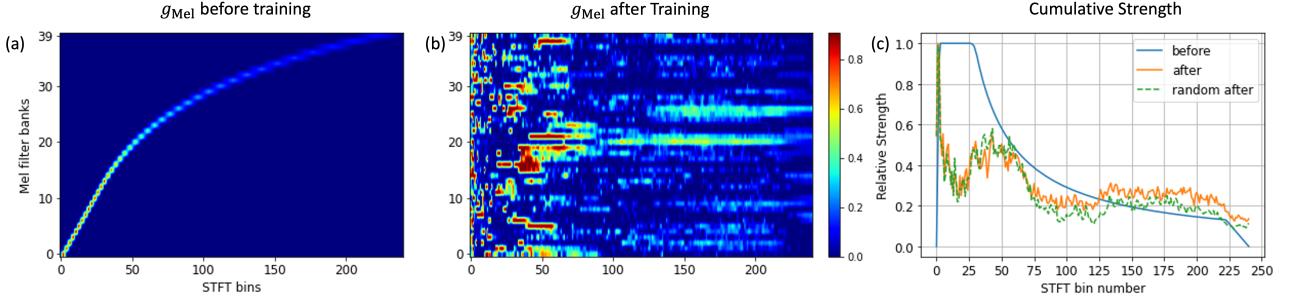


Figure 2: (a)-(b) g_{Mel} before and after training on Google speech commands dataset. (c) shows the cumulative sum of different Mel bases in g_{Mel} .

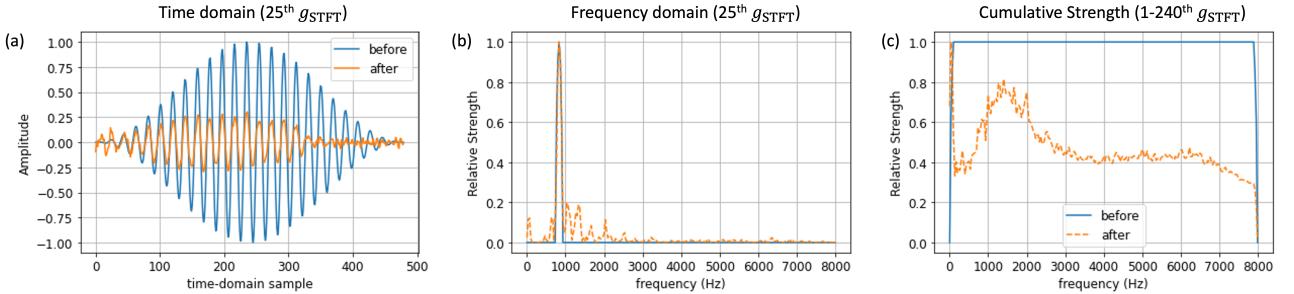


Figure 3: (a)-(b) The 25th STFT basis before and after training (blue and orange lines) on the speech command dataset. (c) Cumulative strengths for 1-240th STFT bases combined.

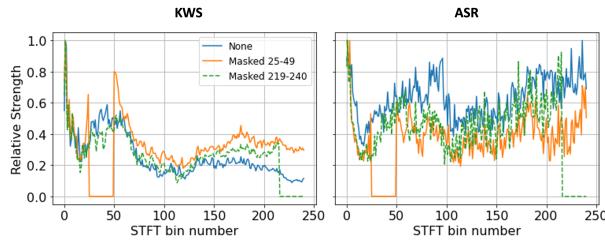


Figure 4: Cumulative strengths when different STFT bins are masked.

information by paying more attention to the remaining available bins. When more bins are masked, for example bins 25-74, the model performance starts to decay. A very similar pattern can be observed for the ASR task.

Based on the experimental results, we believe that trainable basis functions are useful when one of the following two conditions are fulfilled: 1) Model with a limited discriminative power. 2) Input with some missing frequency bins.

4. Conclusions

In this paper, we have shown that trainable STFT or Mel basis functions are helpful when the model capability is limited and some frequency bins are missing from the spectrograms. When these conditions are met, the accuracy for KWS can be improved by up to 14.2 ppt., and the PER can be improved by 9.5 ppt. As the model architecture becomes more mature, trainable basis functions become less effective. They are still useful under a certain circumstances, and can potentially help us bet-

Table 2: Model performance when some frequency bins on STFT are masked by 0. g_{Mel} is either trainable or non-trainable but g_{STFT} is non-trainable in this experiment.

Trainable g_{Mel}	Masked STFT bins	KWS Accuracy	ASR PER
Yes	25-49	45.6%	40.2
	25-74	35.6%	47.3
	216-240	49.5%	38.6
	191-240	46.0%	38.7
	None	42.8%	41.4
No	216-240	18.3%	50.8
	None	36.2%	51.0

ter understand the model’s behaviour. The source code of our models is available online¹.

5. Acknowledgements

This work is supported by A*STAR SING-2018-02-0204 and MOE Tier 2 grant MOE2018-T2-2-161.

6. References

- [1] F. Eyben, M. Wöllmer, and B. Schuller, “Opensmile: the munich versatile and fast open-source audio feature extractor,” in *Proceedings of the 18th ACM international conference on Multimedia*, 2010, pp. 1459–1462.
- [2] ——, “Opener—introducing the munich open-source emotion and affect recognition toolkit,” in *2009 3rd international confer-*

¹<https://github.com/heungky/trainable-STFT-Mel>

- ence on affective computing and intelligent interaction and workshops.* IEEE, 2009, pp. 1–6.
- [3] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, “librosa: Audio and music signal analysis in python,” in *Proceedings of the 14th python in science conference*, vol. 8. Citeseer, 2015, pp. 18–25.
 - [4] D. Bogdanov, N. Wack, E. Gómez Gutiérrez, S. Gulati, H. Boyer, O. Mayor, G. Roma Trepot, J. Salamon, J. R. Zapata González, X. Serra *et al.*, “Essentia: An audio analysis library for music information retrieval,” in *Britto A, Gouyon F, Dixon S, editors. 14th Conference of the International Society for Music Information Retrieval (ISMIR). 2013 Nov 4-8; Curitiba, Brazil.[place unknown]: ISMIR; 2013. p. 493-8.* International Society for Music Information Retrieval (ISMIR), 2013.
 - [5] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, “madmom: a new Python Audio and Music Signal Processing Library,” in *Proceedings of the 24th ACM International Conference on Multimedia*, Amsterdam, The Netherlands, 10 2016, pp. 1174–1178.
 - [6] D. Li, Y. Zhou, Z. Wang, and D. Gao, “Exploiting the potentialities of features for speech emotion recognition,” *Information Sciences*, vol. 548, pp. 328–343, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0020025520309518>
 - [7] K. W. Cheuk, K. Agres, and D. Herremans, “The impact of audio input representations on neural network based music transcription,” in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–6.
 - [8] J. Lee, J. Park, K. L. Kim, and J. Nam, “Samplecnn: End-to-end deep convolutional neural networks using very small filters for music classification,” *Applied Sciences*, vol. 8, no. 1, p. 150, 2018.
 - [9] J. Thickstun, Z. Harchaoui, and S. M. Kakade, “Learning features of music from scratch,” in *International Conference on Learning Representations (ICLR)*, 2017.
 - [10] D. Seo, H.-S. Oh, and Y. Jung, “Wav2kws: Transfer learning from speech representations for keyword spotting,” *IEEE Access*, vol. 9, pp. 80 682–80 691, 2021.
 - [11] J. C. Brown, “Calculation of a constant q spectral transform,” *The Journal of the Acoustical Society of America*, vol. 89, no. 1, pp. 425–434, 1991.
 - [12] V. Tiwari, “Mfcc and its applications in speaker recognition,” *International journal on emerging technologies*, vol. 1, no. 1, pp. 19–22, 2010.
 - [13] A. Berg, M. O’Connor, and M. T. Cruz, “Keyword transformer: A self-attention model for keyword spotting,” *arXiv preprint arXiv:2104.00769*, 2021.
 - [14] S. Majumdar and B. Ginsburg, “Matchboxnet: 1d time-channel separable convolutional neural network architecture for speech commands recognition,” *arXiv preprint arXiv:2004.08531*, 2020.
 - [15] R. Vygon and N. Mikhaylovskiy, “Learning efficient representations for keyword spotting with triplet loss,” in *Speech and Computer*, A. Karpov and R. Potapova, Eds. Cham: Springer International Publishing, 2021, pp. 773–785.
 - [16] D. Coimbra de Andrade, S. Leo, M. Loesener Da Silva Viana, and C. Bernkopf, “A neural attention model for speech command recognition,” *ArXiv e-prints*, Aug. 2018.
 - [17] K. W. Cheuk, H. Anderson, K. Agres, and D. Herremans, “nnaudio: An on-the-fly gpu audio to spectrogram conversion toolbox using 1d convolutional neural networks,” *IEEE Access*, vol. 8, pp. 161 981–162 003, 2020.
 - [18] Q. Fu, Z. Teng, J. White, M. Powell, and D. C. Schmidt, “Fastaudio: A learnable audio front-end for spoof speech detection,” *arXiv preprint arXiv:2109.02774*, 2021.
 - [19] B. Balamurali, K. E. Lin, S. Lui, J.-M. Chen, and D. Herremans, “Toward robust audio spoofing detection: A detailed comparison of traditional and learned features,” *IEEE Access*, vol. 7, pp. 84 229–84 241, 2019.
 - [20] Z. Wu, J. Yamagishi, T. Kinnunen, C. Hanilçi, M. Sahidullah, A. Sizov, N. Evans, M. Todisco, and H. Delgado, “Asvspoof: the automatic speaker verification spoofing and countermeasures challenge,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 4, pp. 588–604, 2017.
 - [21] H. Zeinali, T. Stafylakis, G. Athanasopoulou, J. Rohdin, I. Gkinis, L. Burget, J. Černocký *et al.*, “Detecting spoofing attacks using vgg and sincnet: but-omilia submission to asvspoof 2019 challenge,” *arXiv preprint arXiv:1907.12908*, 2019.
 - [22] B. Kim, S. Chang, J. Lee, and D. Sung, “Broadcasted residual learning for efficient keyword spotting,” *arXiv preprint arXiv:2106.04140*, 2021.
 - [23] Y.-Y. Yang, M. Hira, Z. Ni, A. Chourdia, A. Astafurov, C. Chen, C.-F. Yeh, C. Puhrsch, D. Pollack, D. Genzel *et al.*, “Torchaudio: Building blocks for audio and speech processing,” *arXiv preprint arXiv:2110.15018*, 2021.
 - [24] S. S. Stevens, J. E. Volkmann, and E. B. Newman, “A scale for the measurement of the psychological magnitude pitch,” *The Journal of the Acoustical Society of America*, 1937.
 - [25] S. S. Stevens and J. Volkmann, “The relation of pitch to frequency: A revised scale,” *The American Journal of Psychology*, vol. 53, no. 3, pp. 329–353, 1940. [Online]. Available: <http://www.jstor.org/stable/1417526>
 - [26] G. Fant, *Analys av de svenska konsonantljuden: talets allmänna svängningsstruktur.* LM Ericsson, 1949.
 - [27] W. Koenig, “A new frequency scala for acoustic measurements,” *Bell Lab Rec.*, pp. 299–301, 1949.
 - [28] P. Warden, “Speech commands: A dataset for limited-vocabulary speech recognition,” *arXiv preprint arXiv:1804.03209*, 2018.
 - [29] O. Rybakov, N. Kononenko, N. Subrahmanyam, M. Visontai, and S. Laurenzo, “Streaming keyword spotting on mobile devices,” *arXiv preprint arXiv:2005.06720*, 2020.
 - [30] J. S. Garofolo, “Timit acoustic phonetic continuous speech corpus,” *Linguistic Data Consortium*, 1993, 1993.
 - [31] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks,” in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 369–376.
 - [32] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.