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) 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.

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) $g_{\text{STFT}}$ and Mel basis functions $g_{\text{Mel}}$ 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 $g_{\text{STFT}}$ 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 $g_{\text{STFT}}$ as well as $g_{\text{Mel}}$ on the raw waveform $X_t \in [-1, 1]^T$, resulting in a spectrogram $X_f \in [0, +\infty)T_f \times 40$:

$$\begin{align*}
g &= g_{\text{Mel}} \circ g_{\text{STFT}} \\
X_f &= g(X_t)
\end{align*}$$

(1)

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

Both nnAudio and FastAudio initialise the $g_{\text{Mel}}$ as triangular shapes [22,25,26,27]. One major difference between nnAudio and FastAudio is that all Mel bases $g_{\text{Mel}}$ in FastAudio have the same amplitude, and the bases remains triangular shape in both the initial and training state. In nnAudio on the other hand, the amplitude of the initialised $g_{\text{Mel}}$ decays as the frequency goes up. During the training state, nnAudio allows $g_{\text{Mel}}$ 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 $g_{\text{Mel}}$ and $g_{\text{STFT}}$ are non-trainable.
B. $g_{\text{Mel}}$ is trainable while $g_{\text{STFT}}$ is fixed.
C: $g_{\text{Mel}}$ is fixed while $g_{\text{STFT}}$ is trainable.

D: Both $g_{\text{Mel}}$ and $g_{\text{STFT}}$ 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 $g_{\text{STFT}}$ provided by nnAudio with a Hann window size of 480 samples, and a hop size of 160 samples. For $g_{\text{Mel}}$, 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:

$$Y = (f \circ g)(X_t) = f(X_f)^C,$$  \hfill (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 [23] 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 [23, 24, 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 $1 \times 10^{-3}$. 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 $g_{\text{STFT}}$ and $g_{\text{Mel}}$ with nnAudio results in a higher accuracy. When both $g_{\text{STFT}}$ and $g_{\text{Mel}}$ are trainable (setting D), the accuracy is 14.2 percentage point higher than with fixed $g_{\text{STFT}}$ and $g_{\text{Mel}}$ (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 $1 \times 10^{-5}$ to $1 \times 10^{-3}$) 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 $g_{\text{Mel}}$ 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 \(X_c\).

The shape constraint imposed by FastAudio does not improve the accuracy for this task. Whenever the \(g_{\text{Mel}}\) 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 \(g_{\text{STFT}}\) (setting C) the model performance is relatively unchanged when the number of Mel bases in \(g_{\text{Mel}}\) is reduced from 40 to 30.

### 3.2. Automatic Speech Recognition

The TIMIT dataset [39] 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 \(1 \times 10^{-3}\). 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 \(T\) 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 \(g_{\text{Mel}}\) (setting B) worsens the PER from 31.2 to 33.9. Unlike KWS, trainable \(g_{\text{Mel}}\) (setting B) are more effective than trainable \(g_{\text{STFT}}\) (setting C). When nnAudio Simple is used, the former setting improves the PER from 51.0 to 41.4, and the latter worsens 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 \(g_{\text{Mel}}\) after training on the speech commands dataset in Figure 3(a)-(b). The following discussion is based on Simple as the classifier with trainable \(g_{\text{Mel}}\) and fixed \(g_{\text{STFT}}\).

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 3(c) shows the cumulative importance of different STFT bins before and after training. This is calculated by summing all bases in \(g_{\text{Mel}}\) and then normalizing by the largest value. Before training, the \(g_{\text{Mel}}\) extract information mostly from STFT bins 0-30, and the importance gradually decays for the higher STFT bins. After training, the \(g_{\text{Mel}}\) 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 \(g_{\text{Mel}}\) 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 3(c), which converges to the same pattern as the one initialized as triangular \(g_{\text{Mel}}\) (orange line). We observe a very similar pattern after training on the TIMIT dataset, but \(g_{\text{Mel}}\) 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 \(g_{\text{Mel}}\) pays attention to. When STFT bins are masked by 0, the \(g_{\text{Mel}}\) 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 \(g_{\text{STFT}}\) also pay more attention to lower frequencies than to higher frequencies. Figure 5 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 5(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 5(b). Rather than the original peaks, smaller peaks emerge around it.

Figure 3(c) shows the the cumulative importance of different frequencies. Similar to \(g_{\text{Mel}}\), 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 \(g_{\text{Mel}}\) is that it excels when some frequency bins on the STFT output are missing. Here, we keep STFT fixed and train the \(g_{\text{Mel}}\). 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 \(g_{\text{Mel}}\) 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 \(g_{\text{Mel}}\) 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 5 we try to understand what is happening when some STFT bins are masked out. When bins 25-49 are missing, the trained \(g_{\text{Mel}}\) 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 \(g_{\text{Mel}}\) is able to compensate for the loss of...
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 is still useful under a certain circumstances, and can potentially help us better understand the model’s behaviour. The source code of our models is available online.

### 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] ——, “Openear—introducing the munich open-source emotion and affect recognition toolkit,” in 2009 3rd international conference on Audio-electronic systems, 2009.

---

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