6-2019

End-to-End Music Transcription Using Fine-Tuned Variable-Q Filterbanks

Frank C. Cwitkowitz Jr
fcc6779@rit.edu

End-to-End Music Transcription Using Fine-Tuned Variable-Q Filterbanks

Frank C. Cwitkowitz Jr

End-to-End Music Transcription Using Fine-Tuned Variable-Q Filterbanks

Frank C. Cwitkowitz Jr

June 2019

A Thesis Submitted
in Partial Fulfillment
of the Requirements for the Degree of
Master of Science
in
Computer Engineering

R·I·T | Kate Gleason
College of ENGINEERING

Department of Computer Engineering

End-to-End Music Transcription Using Fine-Tuned Variable-Q Filterbanks

Frank C. Cwitkowitz Jr

Committee Approval:

Acknowledgments

Thank you to Juan Cockburn for supporting me during the bulk of my thesis while still at RIT, Andres Kwasinksi for helping me in preparation for my defense, and Alexander Loui for offering valuable feedback on my research. Thank you to Christopher Kanan for providing me with foundational research experience early on. Thank you to Zhiyao Duan for research advice and for helping me get engaged with the Music Information Retrieval community. Lastly, thank you to my professors, labmates and friends for helping me maintain a positive attitude and consistently encouraging me to do my best.

This thesis is dedicated to my family, who have graciously supported me over the course of my college journey, and did not roll their eyes when I suggested going further.

Abstract

The standard time-frequency representations calculated to serve as features for musical audio may have reached the extent of their effectiveness. General-purpose features such as Mel-Frequency Spectral Coefficients or the Constant-Q Transform, while being pyschoacoustically and musically motivated, may not be optimal for all tasks. As large, comprehensive, and well-annotated musical datasets become increasingly available, the viability of learning from the raw waveform of recordings widens. Deep neural networks have been shown to perform feature extraction and classification jointly. With sufficient data, optimal filters which operate in the time-domain may be learned in place of conventional time-frequency calculations. Since the spectrum of problems studied by the Music Information Retrieval community are vastly different, rather than relying on the fixed frequency support of each bandpass filter within standard transforms, learned time-domain filters may prioritize certain harmonic frequencies and model note behavior differently based on a specific music task. In this work, the time-frequency calculation step of a baseline transcription architecture is replaced with a learned equivalent, initialized with the frequency response of a Variable-Q Transform. The learned replacement is fine-tuned jointly with a baseline architecture for the task of piano transcription, and the resulting filterbanks are visualized and evaluated against the standard transform.

Contents

List of Figures

List of Tables

Acronyms

ADSR	Attack-Decay-Sustain-Release
AMT	Automatic Music Transcription
CNN	Convolutional Neural Network
CQT	Constant-Q Transform
DFT	Discrete Fourier Transform
DL	Deep Learning
ERB	Equivalent Rectangular Bandwidth
HCQT	Harmonic Constant-Q Transform
HVQT	Harmonic Variable-Q Transform

LSTM

long short-term memory units

MFCC

Mel-Frequency Cepstral Coefficients

MFE

Multiple f_0 Estimation

MFSC

Mel-Frequency Spectral Coefficients

MIDI

Musical Instrument Digital Interface

MIR

Music Information Retrieval

STFT

Short-Time Fourier Transform

TFR

Time-Frequency Representation

VQT

Variable-Q Transform

Chapter 1

Introduction

Over the years, the Music Information Retrieval (MIR) community has grown and evolved significantly. In particular, the increasing availability of quality music data has fueled the embrace of more data-driven approaches to MIR tasks. As a result, learning-based architectures have exceeded the performance of classical signal processing approaches [3]. This is especially evident for the more challenging MIR tasks such as Automatic Music Transcription (AMT), where Deep Learning (DL) has seemingly become the default approach [4].

While DL is undoubtedly a promising source of innovation for AMT, the state-of-the-art systems still cannot rival human ability and understanding. The gap widens even further within highly unconstrained environments, where there is no limit on instrumentation, noise, or polyphony within a signal [5]. These limitations are faced by the whole of the MIR community, largely irrespective of the task at hand. This is because they are, at least partly, wrought by the harmonic structure of Western music [6], as well as the variability of music data in general.

The input features to any DL model are of great importance. A model's performance generally depends on the amount of information available at this first stage [3], rather than the degree of its complexity. The MIR community typically employs some Time-Frequency Representation (TFR) of an audio signal as input to classification or estimation systems. This is motivated by the rich musical information that can be inferred from the frequency content of music signals. For instance, the fundamental frequency, denoted f_0, of a harmonic complex tone can be mapped directly to the musical idea of pitch, a perceptual property which enables humans to differentiate between musical notes.

Pitched musical sounds are harmonic, meaning that energy is also concentrated at frequencies which are positive integer multiples of the fundamental. The distribution of this energy is unique and maps to an additional perceptual property known as timbre. Further, in the Western music scale notes are spaced in a geometric fashion, such that the frequency of consecutive notes follows a logarithmic pattern. Given these compounding factors, standard feature sets are highly correlated, and retrieving musical meaning from them is akin to untangling a knot. This is precisely why it becomes difficult to analyze unconstrained polyphonic sound mixtures, and why standard TFRs can be limited as input features for certain tasks.

As the MIR community continues to use DL to optimize the parameters of models for music tasks, it will become necessary to push the envelope of feature learning. Although the commonly deployed TFRs, such as Mel-Frequency Spectral Coefficients (MFSC) or the Constant-Q Transform (CQT), are essentially hand-crafted to better model human hearing or the structure of Western music, it is reasonable to suspect that the features they generate may not be optimal for all music-related tasks. For instance, the instrumentation of a music signal may influence the harmonic frequencies which are more or less descriptive for a task or environment. Additionally, the stock filters used to generate TFRs may be less suitable for capturing different types of musical events, such as the onsets of notes for a specific instrument.

In this work, we expand upon the music analysis pipeline by fine-tuning the time-domain filter weights of a Variable-Q Transform (VQT). The learned transform is evaluated against its standard counterpart for the task of piano transcription. Additional experiments explore the efficacy of learning an audio filterbank from randomly-initialized weights. Visualization of the learned filters and their frequency-domain responses are offered along with a discussion focused on interpretation of the learned filters. The contributions of this thesis are as follows:

1. An investigation into the efficacy of learning filterbanks directly from the raw waveform for the task of piano transcription using a large dataset [1]. The filterbanks are initialized randomly and with the frequency response of a VQT.
2. An open-source filterbank module configurable to mimic the CQT, the VQT, or a 3D harmonic version of either. It can replace the calculation of a TFR and is suitable as front-end to standard DL architectures for audio processing. The weights can be fine-tuned jointly or disjointly with the rest of the model, or they can be left alone to compute the unmodified transforms.
3. An example of the filterbank learning process using a strong baseline model for piano transcription [1, 7]. This example illustrates how one may replace the TFR fed into an acoustic model with a filterbank learned for the task at hand.

Chapter 2

Background and Related Work

Sound is a type of signal transmitted through air pressure oscillations which are generated spatially from vibrations [8]. Music[1] is less precisely defined, but can be thought of as an arrangement of sensation-inducing sounds, each of which varies across dimensions like timbre, pitch, duration, and loudness [9]. The main research problems of the MIR community revolve around the estimation, detection, classification, and understanding of the musical attributes and events within sound. The hope is that solutions to MIR problems will enable music search, indexing, and analysis to reach a level of ease and intuition similar to that of text. The primary focus of this work is to improve contemporary approaches to the task of AMT, which seeks to recover the musical information sufficient to generate a visual representation of the underlying music within a sampled audio signal [10].

2.1 Music Theory

Musical instruments create periodic vibrations when notes are played *e.g.* by plucking a string. The resulting air pressure oscillations are called harmonic complex tones. The energy of these tones is concentrated at harmonic frequencies, positive integer

[1]This work deals exclusively with the Western music framework.

multiples $h \in \mathbb{Z}^+$ of a fundamental frequency f_0:

$$f_h = hf_0 \tag{2.1}$$

In (2.1), $h = 1$ would be used to index the first harmonic frequency, also known as the fundamental f_0, of a complex tone. When a note is played, the unique distribution of energy among harmonic frequencies influences a property known as timbre, which determines how the note sounds. Timbre enables a listener to hear and distinguish between multiple instruments in a mixture. The f_0 of a harmonic complex tone is what characterizes it as having a certain pitch, the perceptual property by which humans can rank notes in ascending or descending order based on frequency. Even though different instruments have a unique sound, the f_0 of identical notes is common across them all.

It is important to stress that although the terms note and pitch are often used interchangeably, it is better to think of a note as having a pitch. In western music, there are 12 pitch classes, denoted by the first seven letters of the alphabet and five sharp ($\sharp$), or flat ($\flat$), variations. These pitch classes are ordered, beginning with C, and map to notes cyclically, repeating after an interval known as an octave. The mapping between pitch and note is one-to-one, whereas the mapping between pitch class and note is one-to-several. Figure 2.1 illustrates how the pitch classes correspond to the standard piano key arrangement.

Each note in the Western scale is associated with a pitch class and an octave. The interval between adjacent pitch classes is referred to as a semitone. This means that there are 12 semitone intervals per octave. The term semitone may also be used to abstract the concept of a note. As stated above, the pitch classes are cyclical, *i.e.* an interval of one semitone starting from $B3$ would lead to $C4$. The f_0 in Hz[2] of any

[2]Hertz enumerate the amount of cycles that occur per second, and are the standard unit of measurement for frequency. Alternatively, there exists an integer measurement scheme for frequency, Musical Instrument Digital Interface (MIDI). This system establishes the MIDI frequency 12 as the

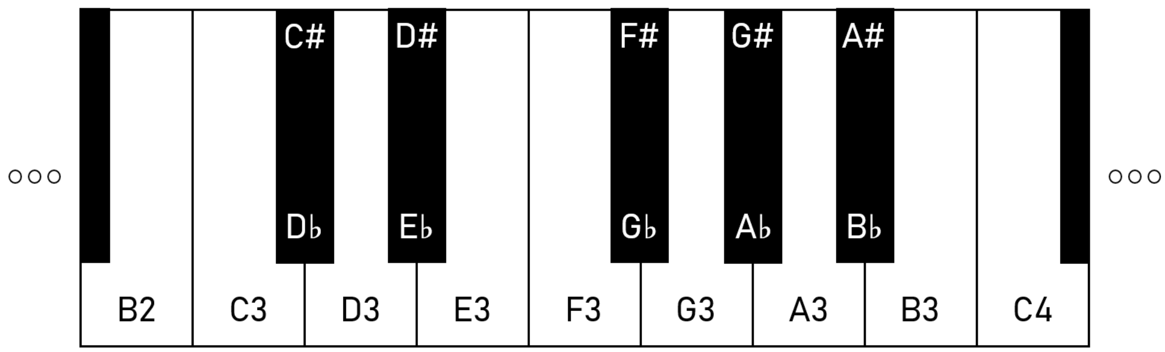

Figure 2.1: Pitch classes as arranged on a keyboard. The white keys map to the first seven letters of the alphabet, whereas the black keys map to the sharp and flat variations. The numbers succeeding each pitch class correspond to the octave containing the notes. A full sized keyboard typically supports the notes $A0$ through $C8$.

semitone n is determined by a geometric relationship which relies on the f_0 of the previous semitone:

$$f_0(n) = 2^{1/12} f_0(n-1) \tag{2.2}$$

In order to establish the relationship in (2.2), the f_0 of the note $A4$ is typically fixed. Notice that the difference in frequency of adjacent semitones over the full set is linear on a log scale scale. One result of (2.2) is the fact that the f_0 of any note is double the f_0 of the note with the same pitch class in the previous octave. For example, if the f_0 of the note $A4$ is fixed at 440 Hz, this makes the f_0 of the note $A5$ 880 Hz, and the f_0 of the note $A3$ 220 Hz. Additionally, setting the f_0 of $A4$ at 440 Hz would make the f_0 of $A\sharp 4$ 466.16 Hz, the f_0 of $A\flat 4$ 415.30 Hz, and so on.

One challenge with analysis in this musical framework is the amount of harmonic overlap that is possible within mixtures of even low polyphony [6]. When notes sound simultaneously, their individual harmonic energy can mix, and the note sources of the energy can become indiscernible. This is to say nothing of *e.g.* duplicate notes and duplicate pitch classes, which may not even be detectable in certain situations, since the set of harmonics for one source may completely contain the contents of the set of harmonics of the other.

f_0 of the note $C0$ and the MIDI frequency 127 as the f_0 of the note $G9$.

2.2 Audio Signals

In order to analyze audio efficiently with computers, it must be recorded and stored digitally. Recording devices take discrete measurements of the air pressure displacement with respect to a steady state, at a fixed periodicity T_s known as the sampling period. During this process, a signal $g(t)$ which is continuous in time is converted into a digital representation $g[k]$ which has discrete time and is indexed by sample number k:

$$g[k] = g(kT_s) \tag{2.3}$$

In (2.3), the sampling period T_s can be represented equivalently by the sampling rate f_s, which is the inverse of T_s, *i.e.* the number of samples taken per second. Harmonic complex tones are of the most interest when analyzing music signals, especially for problems like AMT. Their presence is typically what we would like to infer when analyzing sampled signals. As stated above, harmonic complex tones are air pressure oscillations with frequency components that are positive integer multiples of an f_0:

$$g(t) = \sum_h A_h \sin\left(2\pi f_h t + \phi_h\right) \tag{2.4}$$

Each frequency component is approximately sinusoidal, meaning that it can be described with an amplitude A_h, a frequency f_h, and a phase ϕ_h. Figure 2.2 illustrates an example of the relationship between a pure tone in continuous time and discrete time. (2.3) is a lossy conversion, and as a result, frequency information will be erased if the continuous signal contains frequency components greater than $\frac{f_s}{2}$, also known as the Nyquist frequency. In Figure 2.2, the pure tone oscillates at a rate equal to the Nyquist, and is therefore detectable. The insufficient sampling in Figure 2.3 leads to an undesirable affect called aliasing. Here, the pure tone completes more than half an oscillation in between samples. As a result, there is not enough information in the

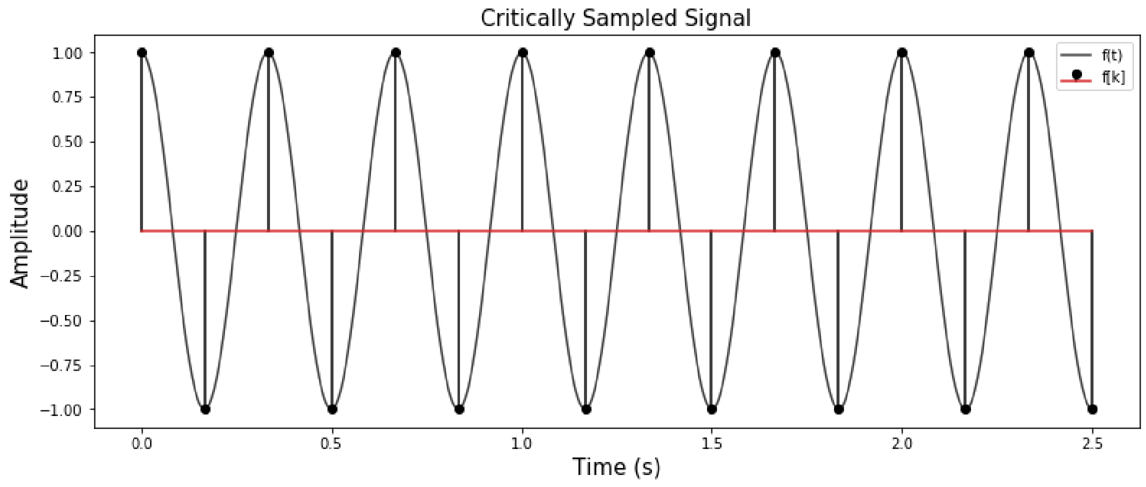

Figure 2.2: Sinusoid oscillating at 3 Hz with a phase of $\frac{\pi}{2}$ and unit amplitude. The signal is being sampled at a rate of $f_s = 6$ Hz.

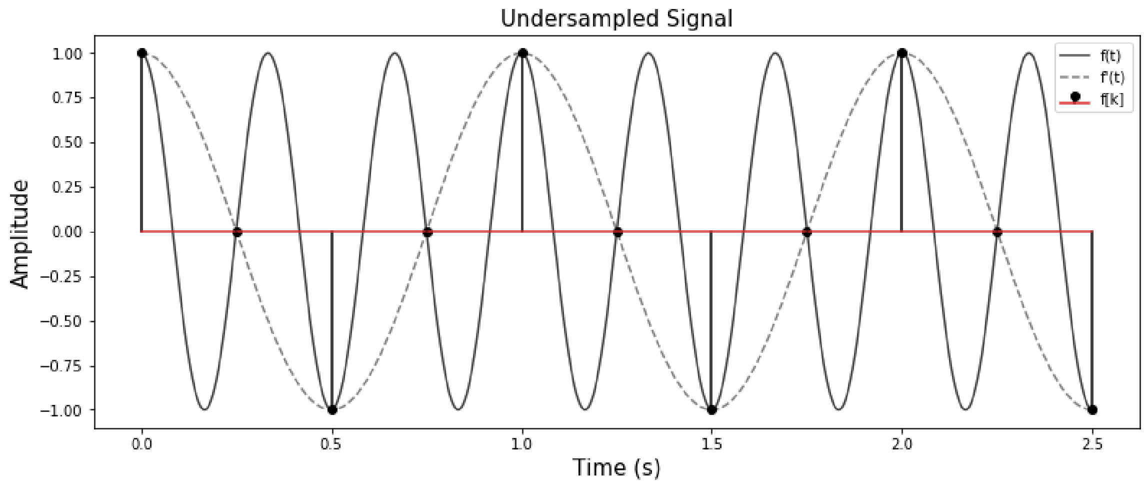

Figure 2.3: Sinusoid oscillating at 3 Hz with a phase of $\frac{\pi}{2}$ and unit amplitude. The signal is being sampled at a rate of $f_s = 4$ Hz, which is not enough for accurate reconstruction. The dotted line illustrates a continuous approximation for the sampled signal.

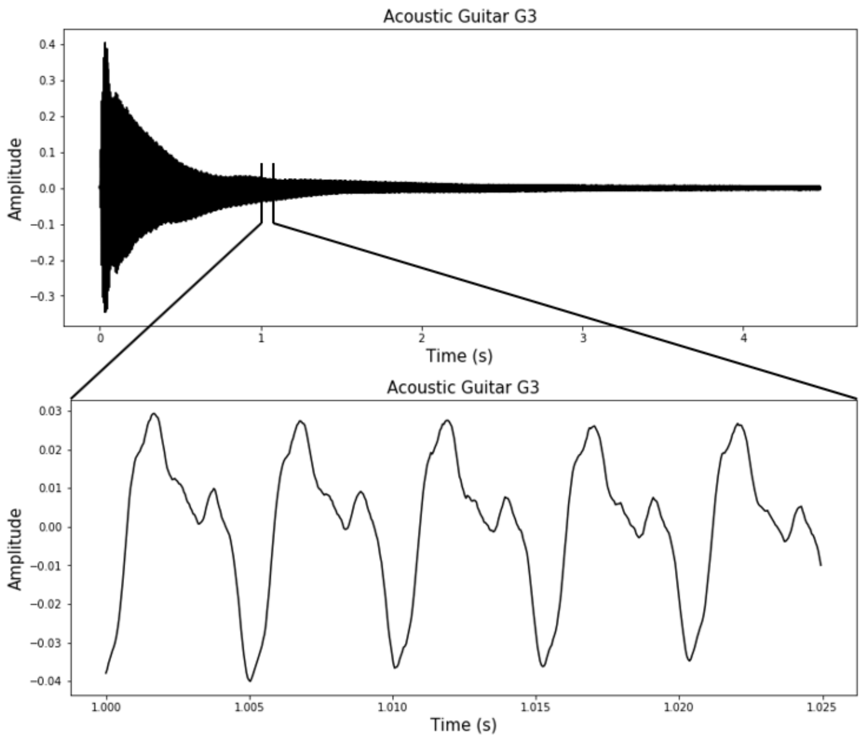

Figure 2.4: Waveform of an acoustic guitar playing the note $G3$, along with a 25 ms closeup of the signal starting at 1 second.

discrete signal to guarantee that there are any frequencies present which are higher than the Nyquist.

The examples in Figures 2.2 and 2.3 are far removed from the complexity of natural music signals. Typically, the complex harmonic tones produced by acoustic instruments are transient and stage-dependent, *e.g.* as detailed by an Attack-Decay-Sustain-Release (ADSR) envelope model [8]. Additionally, the timbre associated with musical instruments necessitates more than just a pure tone oscillating at the f_0. The waveforms associated with two short recordings of an acoustic guitar and an electronic keyboard playing the note $G3$ in Figures 2.4 and 2.5 reveal a more realistic expectation. In both examples, the presence of harmonic complex tones is evidenced by the periodic appearance of the waveform. The lowest periodicity within each occurs roughly every $T_1 \approx 5$ milliseconds. From this, an assertion can be made that $f_0 = \frac{1}{T_1} \approx 200$ Hz. This would be very close to the nominal f_0 for $G3$ under the

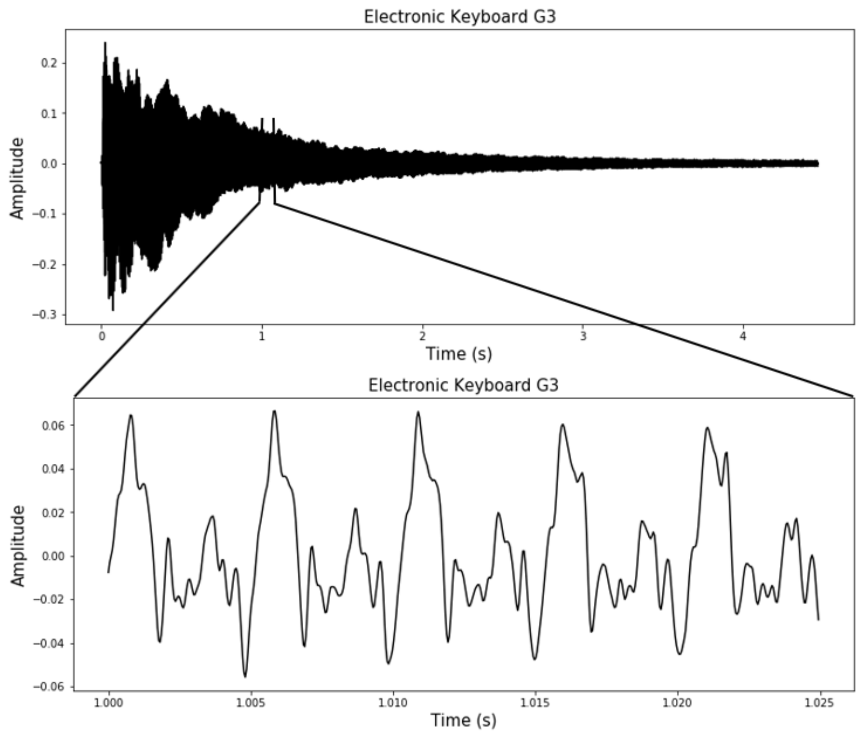

Figure 2.5: Waveform of an electronic keyboard playing the note $G3$, along with a 25 ms closeup of the signal starting at 1 second.

$A4 = 440$ Hz tuning, which would be 196 Hz.

As more challenging tasks like AMT are undertaken, the complexity of the problem only grows. In both Figures 2.4 and 2.5, only one note was being played by one instrument in an environment with little noise. A more characteristic MIR task would be to estimate or analyze the notes present in a recording of a song with multiple instruments, where each is playing multi-note chord arrangements.

2.3 Audio Signal Features

An audio signal is two-dimensional. It merely represents the measured pressure displacement over time [8]. For this reason, the raw signal is not very informative with respect to inferring musical qualities or generating estimates of semantic musical content. This is especially true when the signal being analyzed consists of many harmonic complex tones in addition to stochastic noise. A sampled signal recorded in this environment may appear extremely unintelligible. As mentioned previously, frequency content is of particular interest for AMT and many other music estimation problems. This is because of the close relationship between frequency and pitch, as well as the role of pitch as a fundamental building block of music.

2.3.1 Discrete Fourier Transform

One such set of features from which musical information can be directly inferred is a measure of the degree to which a range of frequencies are present in the signal. This information can be extracted from a discrete signal using the Discrete Fourier Transform (DFT). The DFT transforms a discrete signal into a linear combination of sinusoidal basis functions. The basis functions can be specified entirely by magnitude A_μ, frequency f_μ, and phase ϕ_μ. They are represented as a set of complex coefficients [9]:

$$F(\mu) = \sum_{k=0}^{M-1} g[k] e^{-j\frac{2\pi\mu k}{M}},\ \mu = 0, \dots, M-1 \tag{2.5}$$

(2.5) essentially decomposes the input signal $g[k]$ into a set of frequency components using the complex sinusoids $e^{-j\frac{2\pi\mu k}{M}} = \cos\left(\frac{2\pi\mu k}{M}\right) - j\sin\left(\frac{2\pi\mu k}{M}\right)$. It generates the Fourier coefficients, complex coefficients $F(\mu) = a_\mu + jb_\mu$ for the frequencies within the set $f_\mu = \frac{2\pi\mu}{M}$, where $a_\mu = real(F(\mu))$ and $b_\mu = imag(F(\mu))$. It is important to note the size M of the DFT can be artificially expanded by zero-padding the input signal $g[k]$ for more granular frequency support, but is otherwise typically set to the size of $g[k]$. The absolute value of the complex coefficients $A_\mu = |F(\mu)| = \sqrt{a_\mu^2 + b_\mu^2}$ represents the magnitude of the bases, whereas the angles $\phi_\mu = \arctan\frac{b_\mu}{a_\mu}$ represents the phase.

For many music estimation tasks, the magnitude of the DFT is more informative than the phase. The DFT magnitude can be raised to the second power to generate what is known as the power spectrum. Figures 2.6 and 2.7 represent the magnitude spectrum of the recordings from Figures 2.4 and 2.5, respectively. There are several important attributes of these DFTs to notice. First, in the frequency domain,

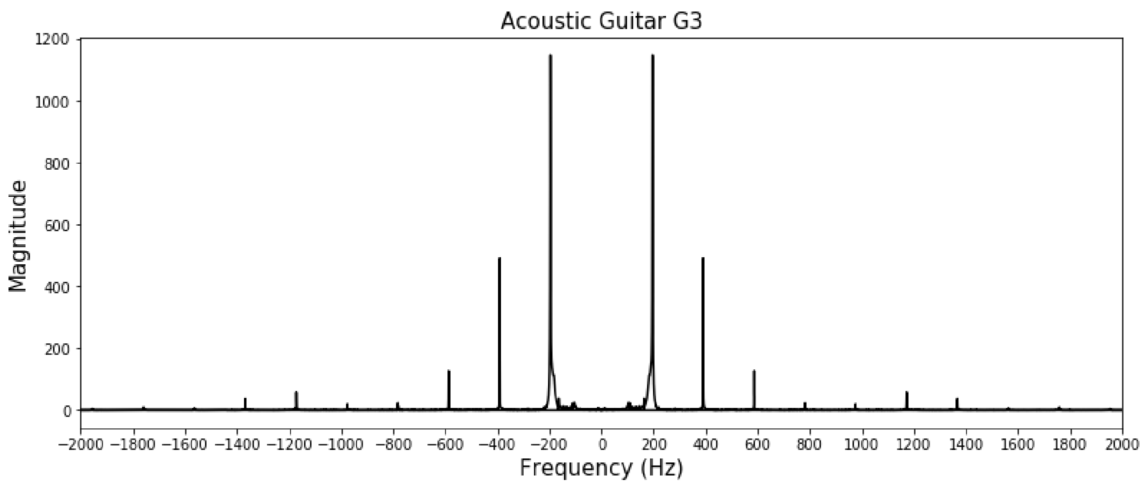

Figure 2.6: Magnitude spectrum of an acoustic guitar playing the note $G3$. There is strong energy at $h = 1$, which corresponds to $\approx$ 196 Hz, less at $h = 2$, even less at $h = 3$, and small but noticeable energy among the other harmonics.

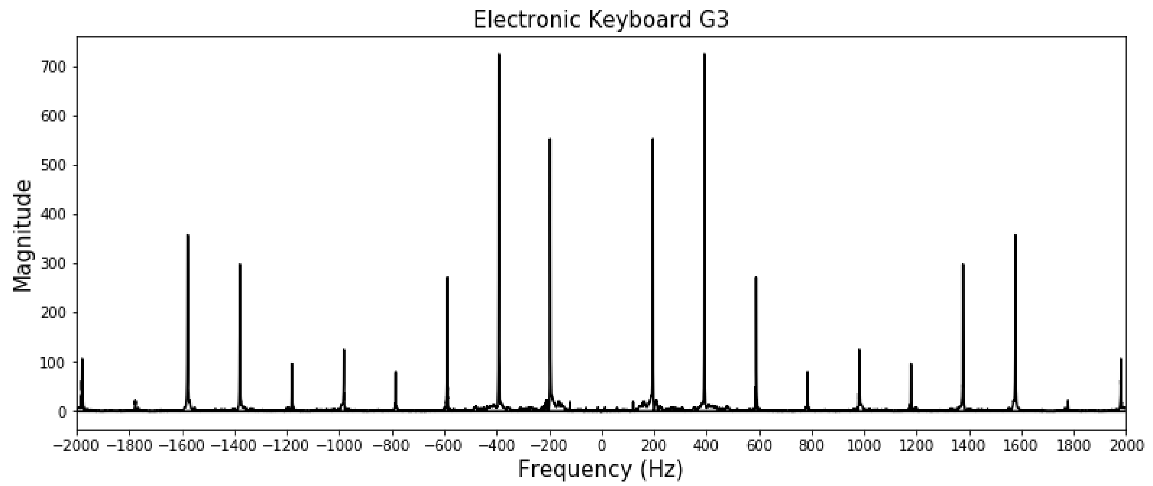

Figure 2.7: Magnitude spectrum of an electronic keyboard playing the note $G3$. There is significant but varying energy at almost every harmonic within what is shown of the DFT, which contains the first ten harmonics.

it is much more obvious where energy is distributed across frequency. Second, it is clear that the complex tones being analyzed are harmonic, *i.e.* all of the frequency components are integer multiples of $f_0 \approx 196$ Hz. Third, the spectral fingerprint or timbre for each instrument is undoubtedly unique. The harmonic energy of the acoustic guitar is not spread out very evenly. Instead, the first harmonic is exceptionally strong, and there is some energy at the second and third harmonic. In contrast, the harmonic energy of the electronic keyboard is much more spread out, and the second harmonic has the most energy. Interestingly, these timbres are unique to the specific characteristics of each physical instrument, not simply the instrument class.

2.3.2 Short-Time Fourier Transform

One drawback of the DFT is that it is cannot specify when each frequency component is active. Rather, it assumes that the frequencies are active for the entirety of the signal. The spectral energy of music signals evolves quickly over time. Timing information corresponding to active frequencies is crucial for a well performing AMT system. A much more powerful set of features for audio signals can be generated by employing the Short-Time Fourier Transform (STFT) [9]:

$$F(\mu, \lambda) = \sum_{k=0}^{M-1} w[k] g[k + \lambda R] e^{-j\frac{2\pi\mu k}{M}}, \ \mu = 0, \ldots, M-1 \tag{2.6}$$

The STFT effectively breaks the analyzed signal into L windowed frames of size M and generates a set of spectral coefficients for each frame, advanced with hop size R and indexed with λ, using the DFT. In (2.6), a window function $w[k]$ is employed to isolate an analysis frame from the rest of the signal. It is also useful in reducing signal activity near the beginning and end of the analysis frame, which will yield spectra with more localized frequency peaks. If a rectangular window function were to be used here, *i.e.* $w[k] = 1 \,\forall\, k$, a phenomena known as spectral leakage would occur.

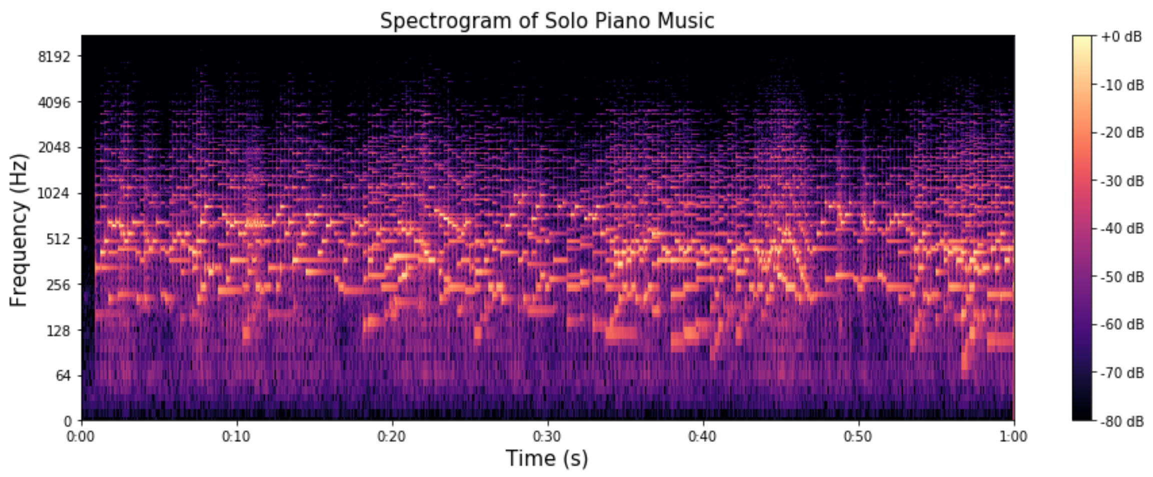

Figure 2.8: Spectrogram for solo piano recording taken from the MAESTRO dataset [1]. The STFT is generated with size $M = 2048$ and hop size $R = 512$.

This can result in de-sparsification of the spectra in order to accurately model the signal, which is undesirable for analysis. A better option would be to use a bell-shaped window function known as the Hanning window:

$$w[k] = 0.5 - 0.5 \cos{(k \frac{2\pi}{M-1})} \tag{2.7}$$

The features resulting from the STFT represent what is known as a TFR, where spectral coefficients vary over time. The resolution in time of a TFR is much more coarse than the sampling period T_s. Instead, there is a set of spectral coefficients for every $T = \frac{R}{f_s}$ seconds. Also notice that the DFT size M determines both the resolution in frequency and in time. Smaller analysis windows are more localized, but can represent less frequencies, and vice versa. A visualization of the STFT output, also known as the spectrogram, for a recording of a solo pianist can be seen in Figure 2.8. The heatmap[3] representation is much more descriptive and visually intuitive than the DFT of the whole signal. The spectral energy changes across time from

[3]Instead of allowing arbitrary amplitudes, as can be seen in Figures 2.6 and 2.7, the log-magnitude $20 * \log(|F(\mu, \lambda)|)$ is a more common way to normalize the spectral energy across the signal using the Decibel scale. In this way, spectral energy is attenuated based on its strength relative to the rest of the signal.

what can be presumed to be a musician playing different notes on the piano.

2.3.3 Mel-Frequency Features

Researchers in the speech domain typically use the psycho-acoustically motivated Mel-scale to generate features which align with the way humans perceive frequency [9]:

$$f_{mel} = 2595 \log(1 + \frac{f_{hz}}{700}) \tag{2.8}$$

Choosing two boundary frequencies in Hertz, one may convert them to Mel and set $N-1$ frequencies linearly between them, before converting these back to Hertz. Then, triangular filters $\Delta_m[\mu]$ centered at the Hertz frequencies f_m can be used to filter the STFT power spectrum in frequency to obtain MFSCs [9]:

$$F_M(m, \lambda) = \sum_{\mu=0}^{M/2} |F(\mu, \lambda)|^2 \Delta_m[\mu], \; m = 0, \ldots, N \tag{2.9}$$

The m^{th} triangular filter is zero outside the range of f_{m-1} and f_{m+1}, and peaks at f_m, where it reaches magnitude one. MFSCs reduce the dimensionality of the STFT spectra and provide a compact set of features which may be better suited for certain audio tasks. The effect of this dimensionality reduction is mostly dependent on the number of Mel frequency bins. As such, they can be equally powerful in settings where music is to be analyzed instead of speech [7, 1].

In Figure 2.9 the MFSCs of the recording from the previous section are calculated and displayed using what is known as the Mel spectrogram, derived from the same Decibel conversion as with the nominal STFT spectrogram. Rather than the linear frequency spacing in Hertz, we can see that the center frequencies of the Mel bins follow a more logarithmic pattern. Additionally, it is clear that energy at higher frequencies is less localized than in the STFT.

Before DL pushed back the envelope of feature processing, many state of the

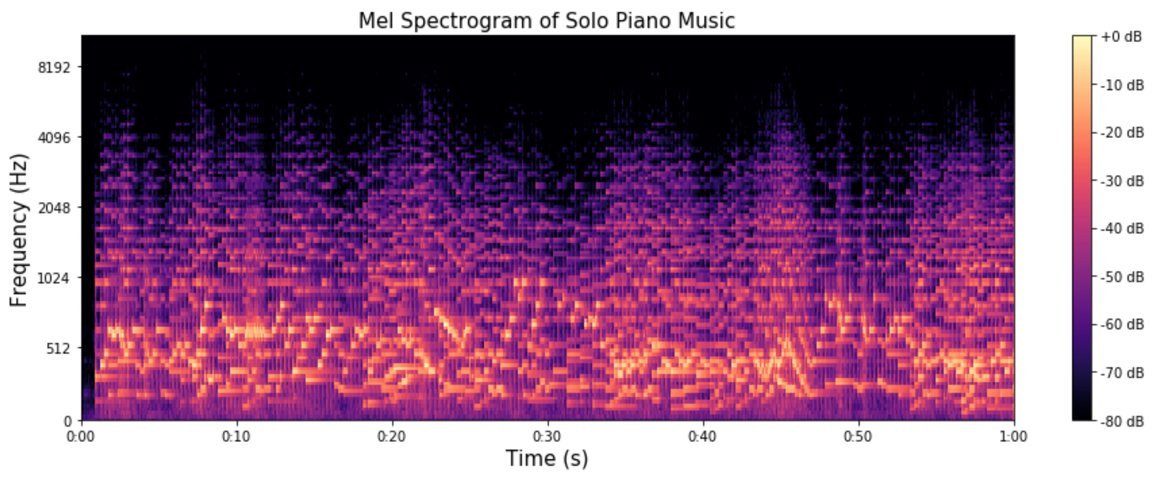

Figure 2.9: Mel spectrogram generated from the STFT in Figure 2.8 using $N = 128$ filters spaced between zero and the Nyquist frequency.

art speech systems operated on further high level features such as Mel-Frequency Cepstral Coefficients (MFCC)s. MFCCs are built atop MFSCs, derived from the Discrete Cosine Transform of the log-scaled MFSCs. After DL, this step became less relevant, as approaches using less meddled-with features were shown to outperform the hand-crafted feature sets [3].

2.3.4 Constant-Q Transform

Although MFSCs are perceptually motivated, they still do not align directly with the content of music signals. In music analysis, the most interesting frequencies for common tasks are those which correspond to the f_0 of music notes. Recall that (2.2) models the f_0 of all notes in the Western music scale, requiring that only one be fixed to instantiate the relationship. Similarly, in the CQT formulation, the Fourier bases are set with center frequencies

$$f_\mu = f_{min} 2^{\frac{\mu}{b}}, \tag{2.10}$$

where f_{min} is chosen as the first center frequency, b defines the number of bins per octave interval, and μ represents the number of semitone intervals beyond f_{min} [11]. The frequency support of every $\frac{b}{12}^{th}$ basis is centered over a semitone, whereas the other bases are spaced linearly between semitones on the log scale. The transform can extend to the Nyquist frequency, or can cease support at a chosen n_{bins} such that $f_{n_{bins}} \leq \frac{f_s}{2}$. With the CQT, rather than using fixed-size frequency analysis windows as in the STFT, a constant Q-factor is maintained for all of the bases. This means that the passband width $B_\mu = f_{\mu+1} - f_\mu$ of the basis associated with each f_μ depends on the constant Q-factor $Q = \frac{f_\mu}{B_u} = \frac{f_\mu}{f_{\mu+1} - f_\mu} = \frac{1}{2^{\frac{1}{b}} - 1}$. As a result, the size in time-domain samples of the analysis frames $M_\mu = Q\frac{f_s}{f_\mu}$ and the necessary windowing function $w_\mu[k]$ differ for each basis [9]:

$$F_C(\mu, \lambda) = \frac{1}{M_\mu} \sum_{k=0}^{M_\mu - 1} w_\mu[k] g[k + \lambda R] e^{-j\frac{2\pi Q k}{M_\mu}}, \ \mu = 0, \ldots, n_{bins} \tag{2.11}$$

If a Hanning window function were used in the case of a CQT, a separate instance would correspond to each basis to match the size of the respective analysis window:

$$w_\mu[k] = 0.5 - 0.5 \cos{(k \frac{2\pi}{M_\mu - 1})} \tag{2.12}$$

Since the size of the analysis frame for each basis is not static, each coefficient is normalized by the analysis frame length M_μ. A CQT spectrogram can be generated as with the previous feature sets. As exemplified in Figure 2.10, the same signal from Figures 2.8 and 2.9 is now much easier to interpret spectrally, especially at lower frequencies. Notice also that the relative spacing between harmonics remains constant for each f_0. For standard CQT representations, there exist efficient implementations that leverage multiplication in the frequency domain instead of convolution as well as iterative octave-wise processing [12]. Signal reconstruction is generally not possible with the CQT, unless it is formulated with redundancy and variable sampling density [13].

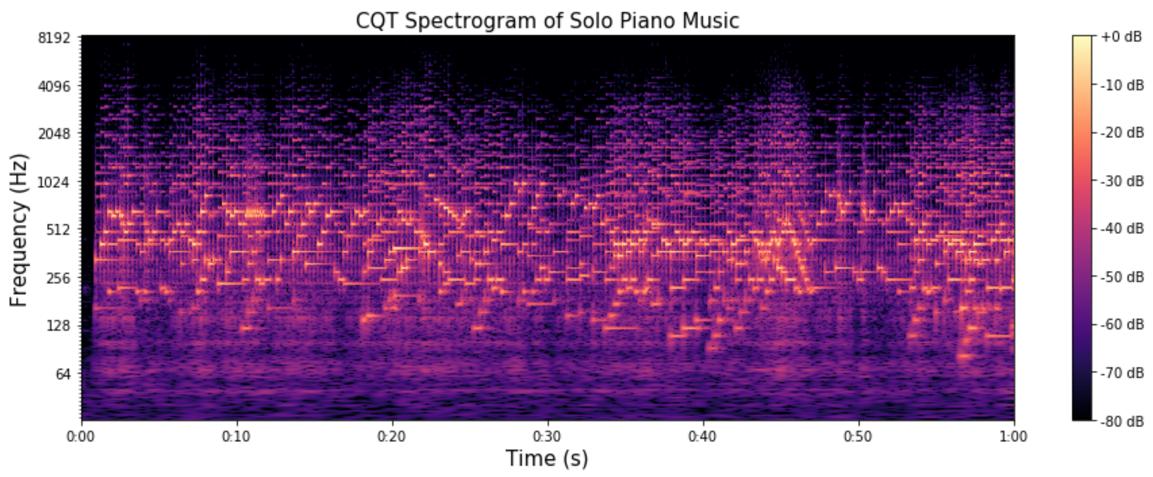

Figure 2.10: CQT spectrogram of the same recording used for Figure 2.8. The CQT is generated starting at $f_{min} \approx 32.7$, spanning 8 octaves with $b = 60$ bins per octave, and advanced with hop size $R = 512$.

2.3.5 Variable-Q Transform

The main advantage of the CQT is its ability to provide the same frequency support to all semitones, and a variable number of bins in between them. Unfortunately, this also has its drawbacks, one being the lack of solid time resolution at lower frequencies. Figure 2.10 clearly shows that higher frequency energy is well localized in time, while lower frequency energy suffers due to compensation for the desired resolution at lower frequencies, where semitones are much less spread out. This tradeoff can be alleviated by introducing another parameter γ to the CQT formulation. The purpose of γ is to dampen the Q factor at lower frequencies, while maintaining roughly a constant Q at higher frequencies [14]:

$$B_\mu = f_{\mu+1} - f_\mu + \gamma \tag{2.13}$$

The additional parameter γ can be interpreted as an offset in Hertz, and is typically chosen to be relatively small, *i.e.* no greater than approximately 30 Hz. Intuitively, γ has a larger relative effect at lower frequencies where the bandwidth is very small, but diminishing effect at higher frequencies. In order to remain musically relevant, the center frequencies do not change. This modification yields what is known as the

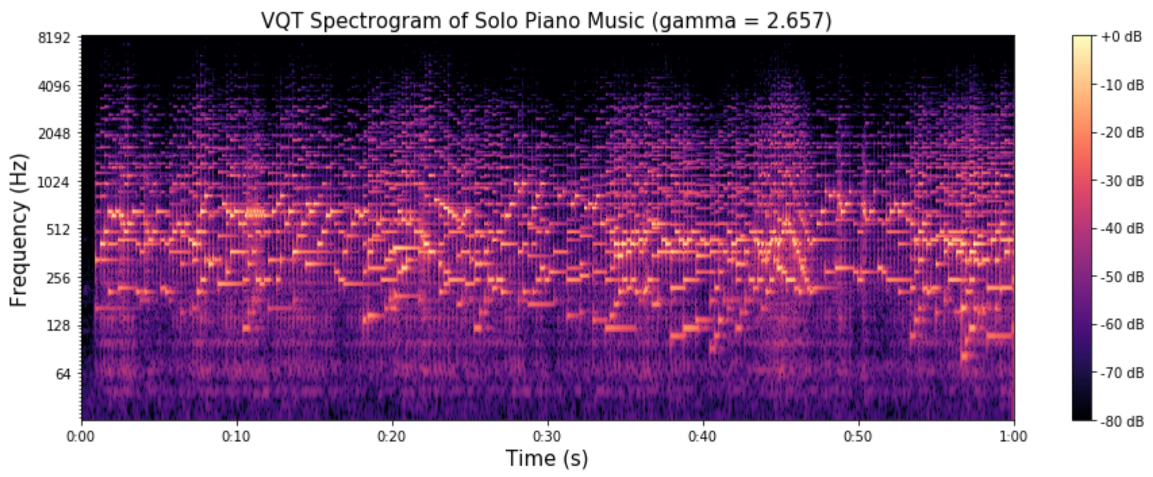

Figure 2.11: VQT spectrogram analysis identical to that of Figure 2.10, but with a smoothly decreasing Q-factor using $\gamma \approx 2.657$. This is the special case where bandwidths vary proportionally to the ERB scale.

VQT.

When $\gamma = 0$, nothing about the formulation in (2.11) is modified, and the transform is identical to the original CQT. The parameter can also be set, such that the bandwidths of the bases are all a constant fraction of the corresponding bandwidths derived from the Equivalent Rectangular Bandwidth (ERB) scale [14, 15]. The ERB scale is psycho-acoustically motivated, and this parameter setting allows the VQT to mimic its bandwidth evolution. This is not to be confused with the actual ERB scale. It can be seen in Figure 2.11 that the lower frequency energy is now much more defined along the time axis. The sacrifice of resolution at higher frequencies is negligible. Note that the lower frequency activity is slightly less discernible along the frequency axis as a result of the VQT. In Figure 2.12 the VQT spectrogram is shown for a larger γ. Although it may seem in this example that too much frequency resolution of lower frequencies has been traded for time resolution, a higher γ allows the size of analysis windows for lower frequency channels to be reduced significantly.

The benefit to applying the VQT over the CQT largely depends on the task at hand. In [16] it was reported that the VQT outperformed the CQT for the task of multi-instrument transcription. The VQT has also gained more traction within the

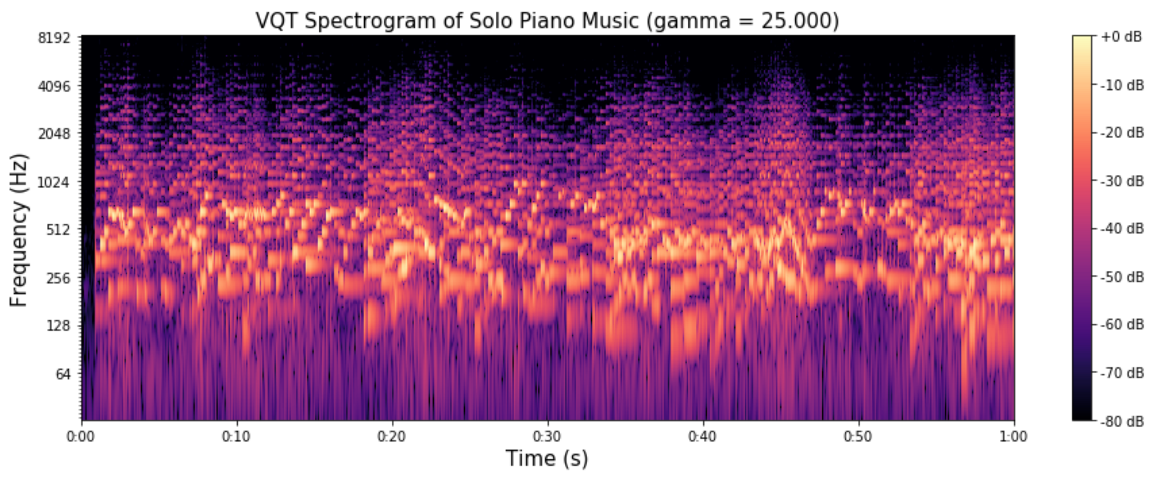

Figure 2.12: VQT spectrogram analysis identical to that of Figure 2.10, but with a smoothly decreasing Q-factor using $\gamma = 25$. The analysis windows toward lower frequencies are shrunk due to the bandwidth offset, and thus, the resolution is poor for lower frequencies, while higher frequency bins still enjoy an approximately constant Q factor.

MIR community in recent years [17, 18, 19, 20].

2.3.6 Harmonic Transforms

The CQT and VQT are suitable for music signals, as the frequency bins map directly to the f_0s of the Western scale. Unfortunately, both transforms are limited in their harmonic coverage of the semitones. Given the center frequencies of (2.10), precise coverage can only be exhibited at power-of-2 harmonics. Clearly, there is no way to produce the integers 3, 5, 6, 7, etc. from any combination of $2^{\frac{m}{n}}$ where m and n are integers. This means that significant harmonic energy finds its way into nearby frequency bins, where it may not be fully aligned, possibly impairing the features for classification systems tasked with *e.g.* AMT. This problem was addressed in [21], where a 3-dimensional TFR known as the Harmonic Constant-Q Transform (HCQT) was constructed by stacking CQTs starting at different harmonics of f_{min}.

$$f_{min}(h) = hf_{min}, \;\; h \in H \tag{2.14}$$

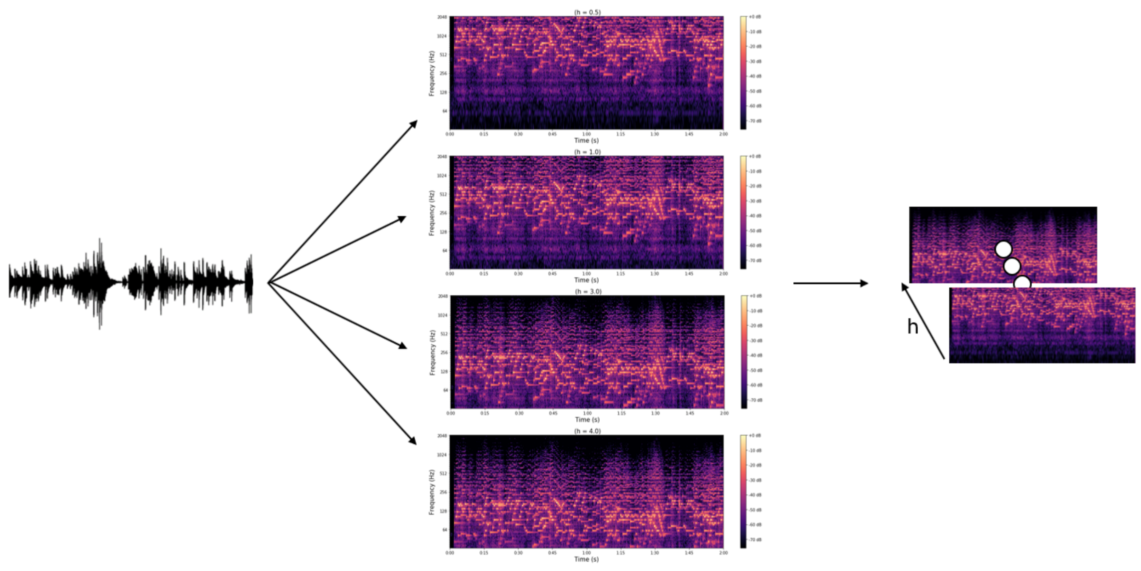

Figure 2.13: Illustration of a harmonic transform corresponding to the VQT calculated for Figure 2.11, where $H = \{0.5, 1.0, 3.0, 4.0\}$. Due to the need for transforms with matching size, only 6 octaves are spanned in the stacked transforms, such that the 4^{th} harmonic transform does not contain filters which exceed the Nyquist frequency in support.

Barring memory and computational constraints, one may choose any subset H of harmonics for which to generate CQTs. Since the only effect of the chosen set is scaling for f_{min}, decimal and sub-harmonics are also possible. A visualization of the harmonic transform for a music excerpt using various harmonics is offered in Figure 2.13. The HCQT is a rather small fine-tuning that may not manifest visually. As such, besides the noticeable frequency shift, the energy does not appear to vary significantly across the harmonic axis. Note that the VQT can also be used in this arrangement to create a Harmonic Variable-Q Transform (HVQT).

2.4 Audio Feature Learning

A neural network is essentially a parametric mapping from one domain, *e.g.* an input space, to another domain, *e.g.* classwise probabilities. Feature learning within a neural network architecture can be viewed as an optimization problem characterized as the minimization of an objective function. The objective function typically aims

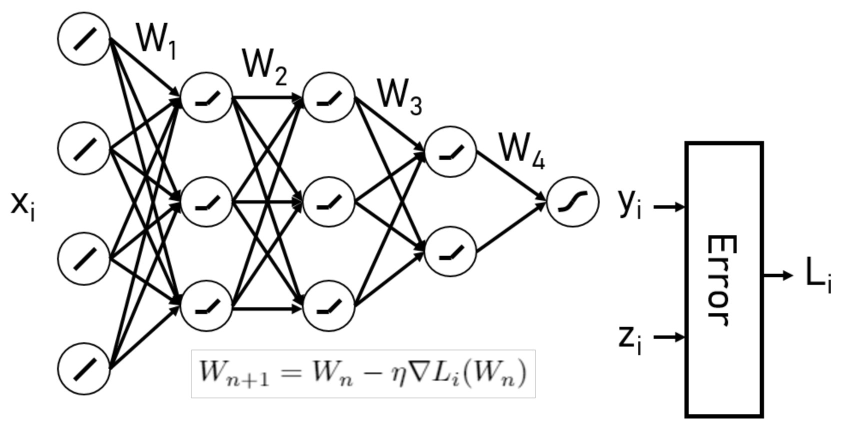

Figure 2.14: Example of multi-layer perceptron with ReLU activation for hidden neurons and sigmoid activation for the output neuron.

to calculate some metric of performance, the loss or error L_i, derived from the relationship between a model's response y_i to some input x_i and the desired response z_i, the ground truth data.

In a supervised learning scenario, a neural network formulation contains many parameters or weights W which are iteratively updated to generate better responses across a dataset, such that some classification or regression goal may be achieved. Weights are initialized randomly and updated by computing the derivative of the loss at a given training step with respect to each parameter of the neural network. The derivatives can be efficiently computed using the back-propagation algorithm and bundled together to form the gradient. In each iteration, where a group of training samples is presented to a model and a loss is calculated, all of the weights are updated to move away from the gradient by a fixed amount η known as the learning rate. This equates to an iterative descent of a stochastic performance surface, which ideally transfers to robust and invariant responses for input samples outside of the training set.

Deep neural networks characteristically include many layers of processing. They are built from various combinations of components including multi-dimensional convolutional layers, activations, feature aggregation and pooling layers, fully connected

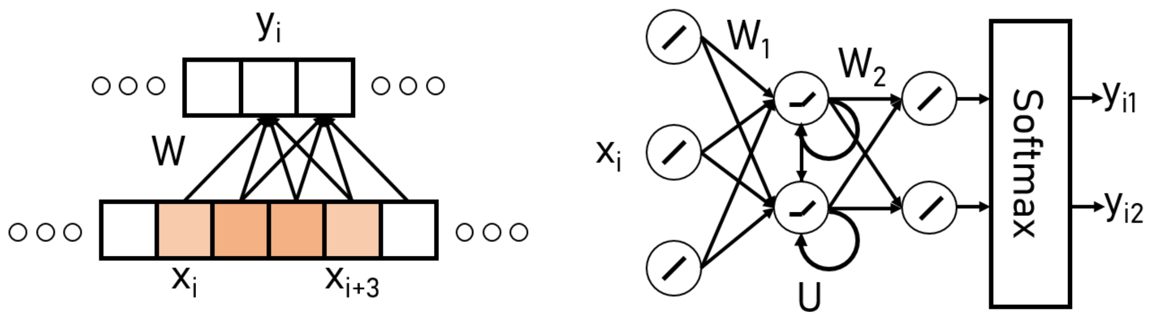

Figure 2.15: Examples of neural network components. (*Left*) 1D convolutional layer with 1 filter of size 4 and a stride of 1. The filter weights are shared across the input signal. (*Right*) Multi-layer perceptron with recurrent weights U and a softmax layer to obtain pseudo-probabilites for mutually exclusive classes 1 and 2.

layers, recurrent layers, etc. There are also plenty of techniques such as batch normalization and dropout which can improve training efficiency. A full overview of DL and more general machine learning techniques is outside of the scope of this work. Excellent surveys of DL as it applies to MIR and other audio can be found in [22] and [23].

After the successes of DL applied to computer vision and speech recognition problems, the MIR community began to increasingly embrace learning architectures for various tasks. DL approaches are well-suited for MIR, since they are able to exploit the hierarchical structure of music [3]. What's more, the traditionally separated feature extraction and classification algorithms are blurred into one within deep neural networks [24], removing the need to iterate endlessly on either. This means that with a well-formulated objective function, sufficient data, and enough computational resources, an optimal model for a task can be found with DL.

Traditionally, the input features that have been fed to deep neural networks for audio tasks involving speech and music have been TFRs or feature sets built upon them. Whether the TFR features are perceptually or musically motivated, they are generated from a bank of stock filters with strictly defined bands centered at fixed frequencies. Although standard TFRs like MFSCs and the CQT are appropriate feature representations for audio, they may not be equally powerful for all tasks from a data-driven perspective. Alternatively, the calculation of a TFR may be replaced with 1D convolutional filters which can be inserted at the front of a deep neural network and learned jointly with the rest of the model. In this approach, learning can be expanded such that the waveform is given as input instead of a TFR. For instance, in Figures 2.6 and 2.7, the timbre of instruments affects the harmonic energy distribution of notes. Similarly, Alvarado et. al. [25] determined that priors were needed to model the frequency content of sounds to be detected after learning spectral templates for Bayesian inference. A deep neural network tasked with *e.g.* AMT might leverage this information while learning to extract features and classify notes in an end-to-end fashion.

Several attempts have been made to learn filterbanks for music related tasks. Dieleman et. al. [26] fed waveform features into a Convolutional Neural Network (CNN) for the task of automatic music tagging, though they found that the model performed better with spectrogram features. Verma et. al. [27] trained a multi-layer perceptron with waveform features for the task of f_0 estimation. Lee et. al. [28] experimented with adding multiple layers below the frame level, effectively adding more depth to the waveform feature extraction process. Perhaps most relevant to this work, [29] trained several models using waveform features for frame-based music transcription, and found that the learnt filters resembled those of an STFT while improving performance.

As a whole, the current trend in MIR still involves using log-scaled TFRs as input features to deep neural networks. This may be due to the need for sufficiently large datasets to properly learn convolutional features competitive with those of typical TFRs for music signals [30]. There are also possible pitfalls of learning end-to-end, such as learned filterbanks overfitting to a dataset or the loss of structure inherent with the frequency-ordered channels of a fixed transform [31]. In order to overcome these challenges in this work, we utilize a large dataset for the task of music transcription, and initialize convolution filters with the frequency response of a standard TFR. In this way, a significant amount of training involves fine-tuning filterbanks rather than realizing them from random weights.

This work is motivated largely by the success of filterbank learning observed within the speech community for several tasks. A variety of approaches have been proposed which span from joint learning of spectrogram filters in place of MFSCs [32, 33], to learning time-domain filterbanks directly from the waveform [34]. In the speech community, filterbank learning directly from the waveform has proven to generate results consistent with Mel-spectrograms [35]. Several studies highlight the importance of low-pass filtering, *e.g.* with pooling layers, and global or instance-wise normalization for quick convergence [36, 37, 38]. In systems where 1-D convolutional filters replace standard TFRs, the filter size, number of filters, and stride are incredibly important with respect to the performance of a learned filterbanks [39]. When learning is successful, resulting filterbanks tend to mimic spectrogram filters that resemble auditory filterbanks [35, 40].

Interestingly, some results suggest that the learned filterbanks are still not superior, and combinations of learned and standard features as input lead to even further improvement [36, 35]. Some systems utilize filterbank learning at multiple scales, aggregating the filter outputs for a final representation [39]. In [41], multichannel time domain filters were learned and shown to be sensitive to frequency and direction of arrival. More recently, [42] formulated MFSCs in time domain using the scattering transform, reaching state of the art performance for a phone recognition task. Several other approaches have added more constraints to the process of filterbank learning, such as the optimization of parametric Gaussian functions for frequency responses [43], or the optimization of low and high cutoff frequencies for a set of bandpass filters [44].

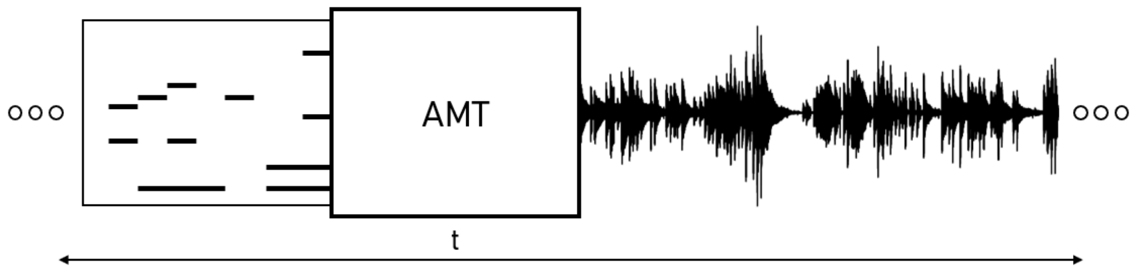

Figure 2.16: Visualization of main idea behind AMT. An audio signal is parsed while semantic musical notation (in this case a piano-roll representation) is created. While this illustration suggests real-time operation and mono-linear signal analysis, neither are hard constraints.

2.5 Automatic Music Transcription

Transcription is the process by which the notation corresponding to music is realized through listening and understanding. It has a broad impact on the landscape of musicians who want to learn how to play the songs from their favorite artists, or those who wish to compose music in real-time. Additionally, the listening and understanding capabilities necessary to transcribe music have other applications, such as real-time instructional music scenarios which listen and provide feedback, mid-level music representations for database querying, and the improvement of approaches to other MIR tasks. Currently, transcription is a skill only experts with extensive music knowledge or experienced musicians possess. Even still, it is an expensive and inefficient process that is prone to human error and does not scale well.

AMT is an MIR task which seeks to solve the transcription problem. That is, AMT approaches algorithmically recover the information sufficient to form a symbolic representation of the music inherent in an audio signal. There are several degrees to how this can be interpreted, each of which expects increasingly articulated symbolic representations. For instance, in Figure 2.16, AMT is portrayed as the extraction of notes from a music signal into a pianoroll representation. In a pianoroll representation, each note is described with an f_0 and a duration. Duration implies that two times,

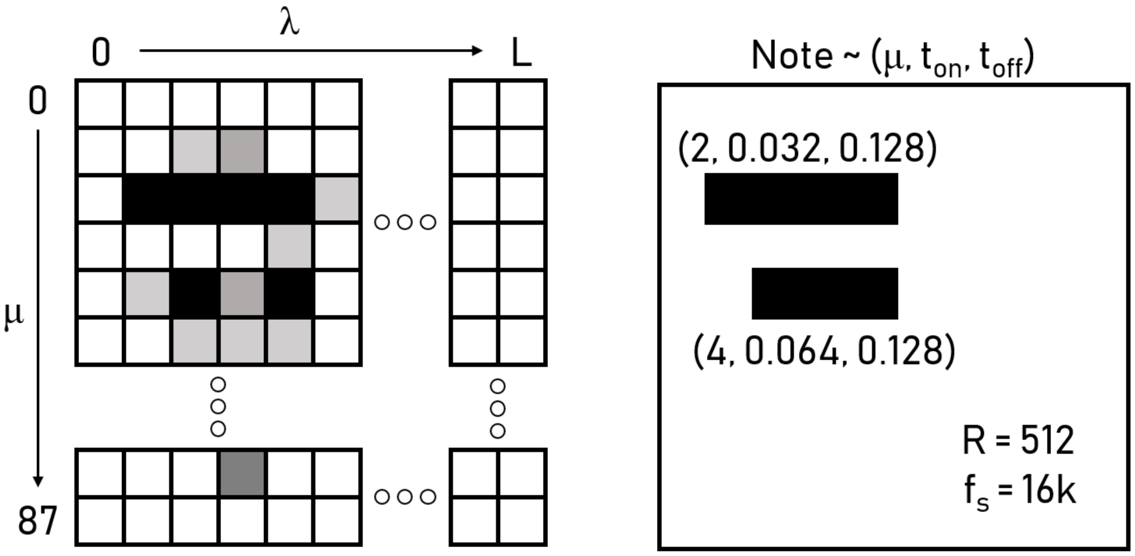

Figure 2.17: (*Left*) Pitch salience representation for a set of musical frequencies, indexed by μ for L frames indexed by λ. (*Right*) Note-level predictions generated using pitch salience refined with *e.g.* onset and offset predictions.

the onset and the offset, are necessary to define a note. There are other more simple forms of musical notation, such as guitar tablature, which do not typically require an offset be defined for notes. Other forms of notation, *e.g.* sheet music, are more descriptive and require additional attributes about a recording. These can include loudness, instrument-specific note segmentation, timing information, etc. Across all notation, regardless of complexity, f_0 activity serves as the foundational ingredient.

The acquisition of active f_0s across time is already challenging enough to merit its own MIR task, Multiple f_0 Estimation (MFE). One could consider MFE a means to generate a frame-based transcription, where estimates are quantized into analysis frames separated by a fixed resolution. Here, the task becomes the mapping of a small audio segment to a set of frequencies. MFE is not specific to MIR, being that it is also relevant from a speech perspective. However, analyzed through an MIR lens, MFE may be interpreted as a frame-wise multi-class binary detection problem, where the classes represent a range of frequencies corresponding to the f_0s of music notes.

For AMT, some form of note tracking is commonly built atop MFE to smooth

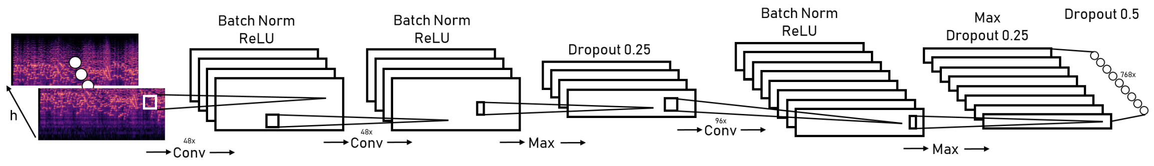

Figure 2.18: Example of an acoustic model for AMT [1], described with more detail in Section 3.1.

predictions and generate note-level predictions rather than frame-level predictions. Note tracking can take different forms, depending on the desired level of transcription complexity for the task at hand. It may include the application of MIR subtasks like onset and offset detection to determine, sometimes ignoring frequency, where notes appear to begin and end. This type of information can further assist in generating predictions when taken into account with frame-wise frequencies. Figure 2.17 provides a visualization of the desired output for each respective task. In [45], a neural network with several different training configurations was applied to MFE. Another MFE approach employed a CNN to learn pitch salience representations, similar to piano rolls, using a large dataset [21].

Overall, AMT is a large multi-faceted task which is challenging due to its complexity, aforementioned harmonic overlap, large degrees of polyphony, variance in recording scenarios and instrumentation, lack of data, noise, etc. For monophonic signals, the story is quite different, and the best approaches are now relatively robust [46]. Before DL was a viable means to tackle AMT, approaches relied upon classic statistical signal processing, heuristics, and shallow classification algorithms [47], which unsurprisingly reached a limit in terms of effectiveness. A good review of this period of AMT history can be found in [5], where authors suggested that approaches can be improved by making them instrument specific, acquiring larger datasets, and jointly tackling other MIR tasks. A more recent review after the passing of several years of DL research for music is presented in [4].

There have been several data-driven neural network approaches to AMT, some

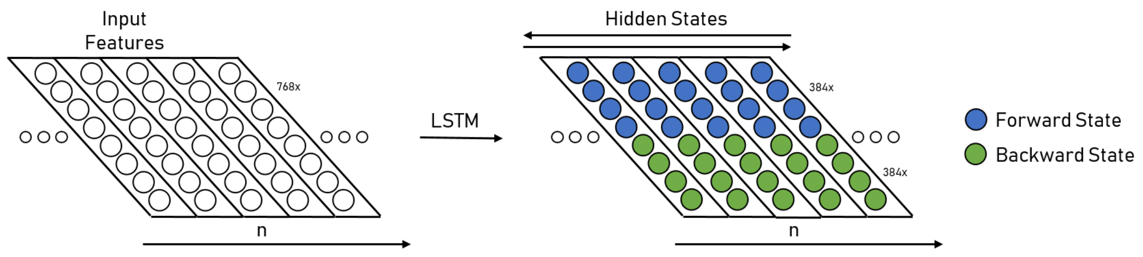

Figure 2.19: Example of a language model for AMT [1], described with more detail in Section 3.1.

involving only recurrent neural networks [48], and some using only convolution [49]. A common formulation involves training both an acoustic model to predict f_0s and a music language model to aggregate the predictions over time for a task [50, 51]. Of this flavor, [7] trained acoustic models jointly for framewise transcription, onset detection, and velocity estimation, where frame predictions were gated by the onset detector, and a language model was incorporated to smooth output. A subsequent iteration of the model was presented in [1], where several improvements were made including the use of separate acoustic models for offset detection and velocity estimation, and a dataset one order of magnitude larger than the previous standard for training. Elowsson [19] trained a layered neural network which performed each subtask jointly in an iterative manner. Kelz et. al. [52] were the first to propose the training of a separate acoustic model for offset detection. Although DL is only one approach to AMT, the rest of the literature is too vast to cover in this work.

Chapter 3

Method

Our approach involves the fine-tuning of a front-end filterbank which accepts a waveform as input. The filterbank replaces the calculation of the TFR corresponding to a waveform for input features. It is implemented as a module[1] configurable such that it can be randomly initialized, or set to match the frequency response of a CQT, a VQT, or a 3D harmonic version of either. When the module is not trained and no other modifications are made, it acts as a convolutional implementation of the configured transform, matching indistinguishably the output from the community standard Librosa implementation[2] [53]. When it is trained, the weights making up the filterbank are allowed to fluctuate based on the gradients of the downstream loss. With the learnable replacement, any arbitrary MIR system can be trained end-to-end and deployed on the raw waveform of music signals. Ideally, fine-tuning the filterbank will lead to the emphasis of relevant harmonic frequencies over less discriminative frequencies for a given task. Another possibility of learning or fine-tuning the filterbank is one which can better understand and capture the ADSR behavior of specific instruments.

In this work, we are not interested in developing a new transcription model, but rather testing out the usefulness of the learnable filterbank. As such, in order to

[1]Our code is written in PyTorch and publicly available at `https://www.github.com/cwitkowitz/LHVQT`.

[2]The Librosa CQT calculation has been modified to incorporate γ for VQT calculation.

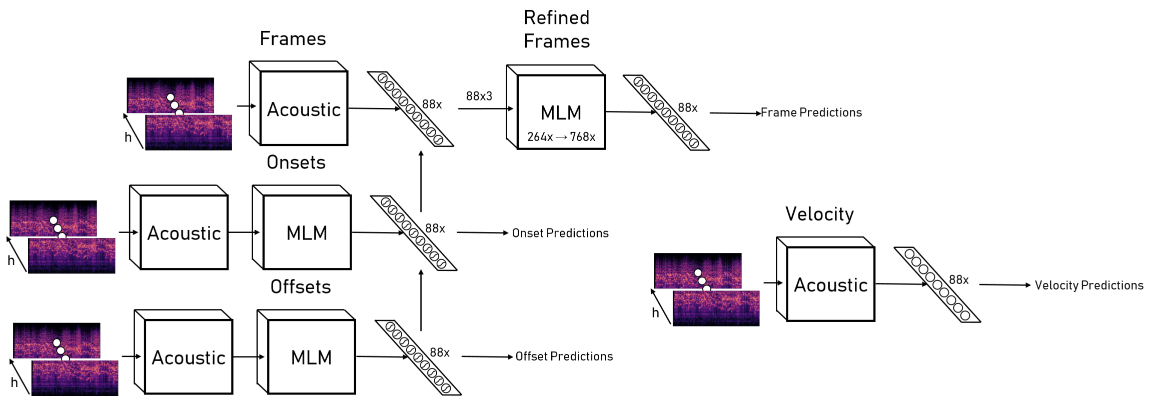

Figure 3.1: Baseline model presented in [1] and implemented in [2]. The specification for the acoustic models and music language models are displayed in Figures 2.18 and 2.19, respectively.

investigate the efficacy of the learnable replacement, we exchange it with the MFSC calculation of a baseline architecture and optimize the combination jointly for the task of piano transcription. These tasks allows us to observe the types of impulse responses and frequency bands learned for transcription tasks with one instrument in isolation.

3.1 Baseline Architecture

The back-end architecture we use for our experiments is a strong baseline developed in [7], improved in [1], and implemented in PyTorch [54] in a publicly available GitHub repository [2]. The model takes as input a TFR with a fixed number of frequency bins and arbitrary length in time. As we will see, this is perfectly compatible with the output of our learned filterbank. The system outputs frame-wise predictions corresponding to onset, offset, and f_0 probabilities as well as velocity estimates for a note range of size $L = 88$.

The entire architecture is illustrated at high-level in Figure 3.1. It is made up of four separate convolutional acoustic models, each intended to perform a different subtask of AMT. These include onset detection, offset detection, f_0 estimation, and velocity estimation. Music language models containing bilinear long short-term memory units (LSTM)s are employed at several stages of processing. The onset and offset acoustic models feed directly into bilinear LSTMs, and each model contains a fully connected layer with L output sigmoid activated units for obtaining pseudo-probabilities. A sigmoid activation is not used for the velocity estimation model, since it is performing regression rather than classification. For a refined f_0 estimation, the initial f_0 predictions are appended to the onset and offset predictions, and fed to an additional bilinear LSTM. The refined predictions are post-processed with a sigmoid activated fully-connected layer.

The convolutional architecture is constant across all acoustic models. It includes three convolutional layers with $\frac{L}{16}$, $\frac{L}{16}$, and $\frac{L}{8}$ learned filters respectively of kernel size 3 by 3. After each convolutional layer, batch normalization and ReLU activation are applied. Before and after the third layer, a max-pooling operation is performed to downsample features along the frequency axis by a factor of two. These are followed by dropout with probability 0.25. Finally, a fully-connected layer with L output units and dropout with probability 0.5 generate the unsmoothed output of the acoustic model. The acoustic model is presented in Figure 2.18.

In order to generate note predictions, several post-processing steps are necessary. First, the frame-wise onset and f_0 pseudo-probabilities are used to generate binary labels using a threshold of 0.5. Pairs of subsequent non-active-to-active onset prediction frames are taken as note candidates. For each candidate, starting from the frame where the prediction became active, if there is either an onset or f_0 activation in the next frame for the candidate, the candidate becomes a note prediction. The predicted onset frame of the note is the first frame where the onset prediction became active. The predicted offset frame of the note is the last consecutive frame where either an onset or f_0 activation exists. The onset and offset frame numbers are converted to onset and offset times. Finally, a note's predicted velocity is taken to be the mean of all velocity predictions for frames with onset activations.

3.2 Learnable Filterbank

In our scheme, we construct a bank of complex-valued time-domain filters with a 1-D convolutional layer. The filters can be initialized such that their center frequencies f_μ and bandwidths M_μ in the frequency domain match a standard filterbank. Our module is configurable through several parameters, of which the most important are explained next.

3.2.1 Parameters

- f_s : sampling rate - necessary to determine the size of each filter M_μ in samples such that the Q-factor Q_μ of the filter is met.
- R : hop length - chosen separation in samples for each frame calculation. In our convolutional implementation, it is the same thing as stride.
- f_{min} : minimum center frequency - required to fix the rest of the center frequencies to geometrically align with music notes. For example, to start the transform at the note C1, $f_{min} \approx 32.7$ Hz.
- b : number of bins per octave - influences the frequency resolution of the transform as well as the amount of features for each semitone.
- n_{bins} : number of frequency bins - effectively determines the maximum frequency of the transform.
- γ : variable-Q parameter - degree of smooth decreasing for the Q-factor towards lower frequencies. This can be set to zero to initialize CQT weights.

- M_{mp} : max pooling amount - max pooling downsampling factor across the time dimension. This can increase invariance to phase or can be set to one to negate the max pooling operation.
- H : set of harmonics - outlines the transforms calculated using (2.14), holding all other parameters constant. If H only contains the first harmonic, *i.e.* $H = \{1\}$, a normal CQT or VQT representation is generated.
- *random* : boolean - after constructing a filterbank of appropriate size for the chosen transform, $random = False$ will load the appropriate weights, whereas $random = True$ will leave the weights randomly initialized, though they are normalized to have a summed magnitude equal to one.

3.2.2 Initialization

In order to mimic the nominal transforms in such a way that they can be fine-tuned, the complex weights of the configured transform must first be calculated. First, the geometric center frequencies for the filters $\mu = 0, \ldots, n_{bins}$ are calculated using (2.10). After the center frequencies have been fixed, the Q-factors for the corresponding filters are calculated with the chosen b using

$$Q_\mu = \frac{f_\mu}{(2^{\frac{1}{b}} - 1)f_\mu + \gamma}. \tag{3.1}$$

If the parameter γ is equal to zero, then 3.1 will produce the same Q factor for each filter, as is the case for the CQT. With the Q factors, the size in samples of each filter can be determined using

$$M_\mu = Q_\mu \frac{f_s}{f_\mu}. \tag{3.2}$$

Once the lengths for the filters have been determined, they are rounded to the nearest integer, and the filter weights W are generated as windowed sinusoids using

$$W_\mu[k] = w_\mu[k]e^{j\frac{2\pi f_\mu k}{f_s}},\ k = -\frac{M_\mu}{2}, \dots, \frac{M_\mu}{2}, \tag{3.3}$$

where the choice of window function w_μ is arbitrary as long as its length coincides with that of the filter. We use the Hanning window function defined in 2.12. The weights are normalized such that their summed magnitude is one. In order to place all of the filter weights in a 1D convolutional layer, they are zero-padded so that they all have the same length M_0 as the largest filter. At this point, the filter bases have been created, but their weights are complex and cannot be supported within PyTorch modules as such. For this reason, a 1D convolutional layer is created with $2n_{bins}$ filters of kernel size M_0, stride R, and padding $\frac{M_0}{2}$. Lastly, if $random = False$, all of the filter weights are split into real and imaginary parts and loaded into the convolutional layer. If $random = true$, the filter weights are discarded, and the weights which were randomly initialized with the instantiation of the convolutional layer remain after undergoing the normalization procedure. Figure 3.2 illustrates how the largest and smallest 1D time-domain convolutional filter appear for a VQT-initialized filterbank where $\gamma = 25$. As can be seen, the real and imaginary parts are slightly out of phase, and the receptive field becomes smaller across frequency channel.

The separation of the weights into real and imaginary produce disjoint transforms F_{real} and F_{imag} which correspond to the real and imaginary parts of the spectrum, respectively. Figure 3.3 shows the layout for the module. Additionally for post-processing, L2 pooling, max pooling, and 1D batch normalization layers are initialized. Their purpose is described further in Section 3.2.3. Since our module is formulated as a harmonic transform, this initialization procedure is conducted separately for a filterbank corresponding to each harmonic in H.

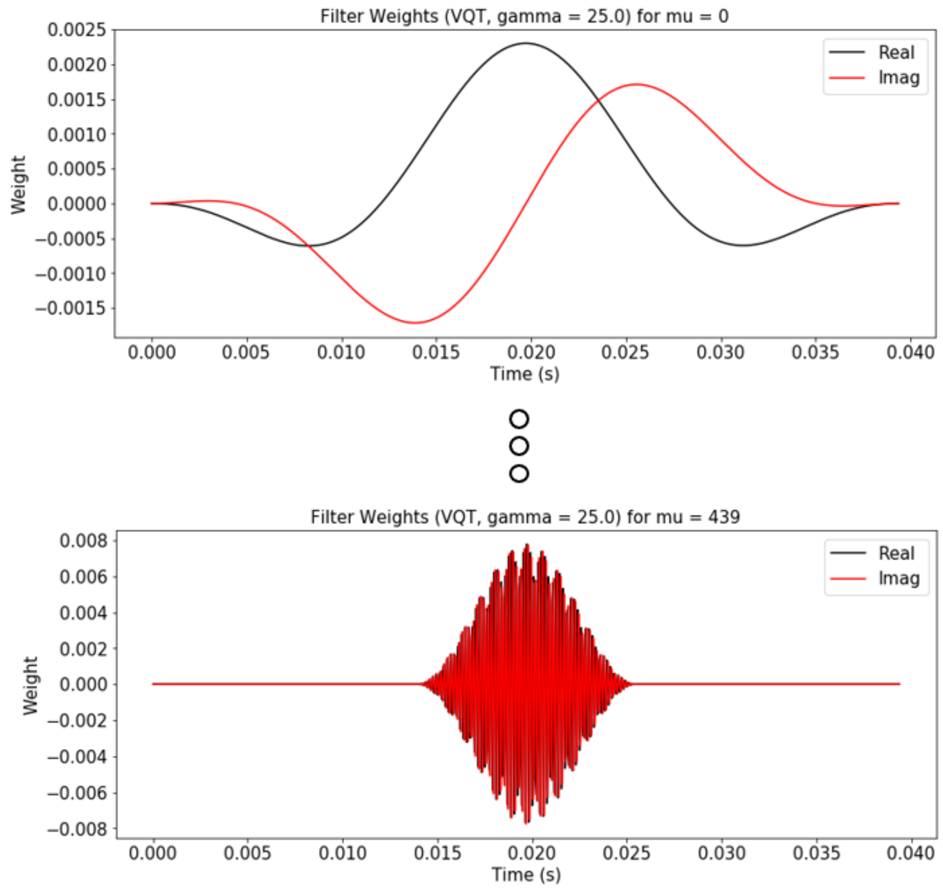

Figure 3.2: Real and imaginary weights for smallest ($\mu = 0$) and largest ($\mu = 439$) filters in the learnable filterbank when $\gamma = 25$.

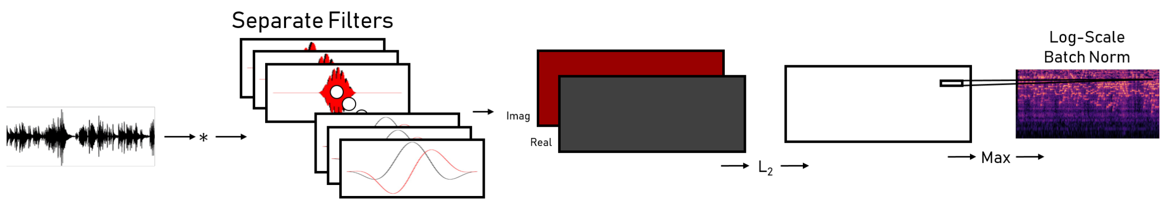

Figure 3.3: Learnable filterbank module constructed for each harmonic within the set of the desired configuration. The stack of output TFRs assembled using each instance of the module is used as input to the baseline transcription model.

3.2.3 Forward Pass

When a waveform is passed to the learnable module, it is transformed with each filterbank in isolation to create 2D TFRs which are later concatenated along a third dimension. For each harmonic filterbank, the filters are convolved with the input signal to produce F_{real} and F_{imag}. In practice, the convolutional filters are interleaved such that adjacent filters make up the real and imaginary components to filterbank filters μ. Thus, the output of the convolutional layer is really $F_{real+imag}$, where $F_{real+imag}(\lfloor \frac{n}{2} \rfloor) = F_{real}(\mu)$ and $F_{real+imag}(\lfloor \frac{n}{2} \rfloor + 1) = F_{imag}(\mu)$ for $n = 0, \ldots, 2n_{bins}$. The magnitude coefficients F are generated by passing $F_{real+imag}$ through an $L2$ pooling layer with a kernel size and stride of 2, effectively computing $F = \sqrt{F_{real}^2 + F_{imag}^2}$. Next, if the filterbanks are not being trained, the output of each filter $F(\mu)$ is scaled by the filter length $\frac{1}{M_\mu}$, as in (2.11). If the filterbank is being trained, this operation is no longer valid, since previously null weights are allowed to fluctuate. The filterbank output is max pooled across the time axis with kernel size and stride M_{mp}. Finally, the filterbank output is log-scaled and passed through a per-channel batch normalization module. Although max pooling and batch normalization are not part of the standard TFR equivalents, they are still used when emulating these fixed transforms.

Chapter 4

Experiments

In order to determine if the learned filters improve transcription, we conduct experiments using several different filterbank configurations on two types of instrument-specific datasets. We connect the learnable filterbank to the baseline architecture, removing the previous calculation of MFSCs. The experimental configurations are outlined in Table 4.1. In experiments 1 through 3, the filterbank weights were fixed and only the baseline architecture was trained. In experiments 4 and 5, the filterbank weights were trained jointly with the baseline architecture. In experiments 6 and 7, the previous two experiments were repeated with a smaller batch size and a larger sequence length.

For each experiment, the convolutional layer was designed such that it could be loaded with the appropriate weights for a filterbank. The appropriate weights were either loaded into the module, or the random weights initialized with the convolution

1.	Fixed Random Weights (FRW)
2.	Fixed Constant-Q Filterbank (FCQ)
3.	Fixed Variable-Q Filterbank (FVQ)
4.	Trained Random Weights (TRW)
5.	Fine-Tuned Variable-Q Filterbank (TVQ)
6.	Trained Random Weights (TRW2)
7.	Fine-Tuned Variable-Q Filterbank (TVQ2)

Table 4.1: Experiment specification and labels.

layer were kept. The convolutional layer design and the loading of the appropriate weights were separated, in order to vary the size of a random-weight filterbank through the VQT parameter γ. In every experiment, the parameters were set such that $f_{min} = f_{C1}$, $n_{bins} = 440$, and $b = 60$. This way, the filterbank initialization covered 88 semitones starting at the note C1, with a resolution of 5 frequency bins per semitone. For all filterbanks, a hop length of $R = \frac{0.032 f_s}{M_{mp}}$ is used as the convolutional stride, where $M_{mp} = 16$. In all experiments except for experiment 2, γ is set to be 25.

Notice that there is no experiment where a CQT filterbank is fine-tuned. This is because the length M_0 of the largest filter within the CQT filterbank becomes extremely large with the desired frequency resolution of $b = 60$. All 1D convolutional filters are zero-padded to match the length of the longest filter and there are too many parameters to learn in a convolution layer of that size. With the parameters specified above, excluding the zero γ necessary for a CQT, the length of the longest filter M_0 is 42110. In this case the analysis window spans ≈ 2.6 seconds for audio sampled at 16000 Hz and signal stationarity assumptions are likely violated.

4.1 Datasets

4.1.1 MAESTRO

The MAESTRO dataset [1] is used for training, validation and testing. It contains approximately 172 hours of virtuoso piano performances played on Yamaha Disklaviers by different contestants of the International Piano-e-Competition. The performances are recorded both acoustically and with a MIDI system. The note-wise annotations are encoded with MIDI and programmatically aligned with the acoustic recordings. We choose data splits which are consistent with those recommended by the authors. The splits are organized such that no performance is repeated outside of a split and the training, testing, and validation portions make up roughly 80%/10%/10% of the data.

4.1.2 MAPS

The MAPS dataset [55] is used for additional testing. It includes, among other content, synthesized renderings and real recordings of full-piece music performances. We evaluate our approach using only the real recordings, and apply the same pre-processing as in [7]. The purpose of evaluation on MAPS is mainly to offer a point of comparison between the results of our approach and previous approaches. This was the primary piano transcription dataset before the recent release of MAESTRO.

4.2 Metrics

Relevant metrics for AMT include precision pr, recall re, and F_1 score across K total predictions [56]:

$$pr = \frac{\sum_{k=0}^{K-1} TP[k]}{\sum_{k=0}^{K-1} TP[k] + FP[k]} \tag{4.1}$$

$$re = \frac{\sum_{k=0}^{K-1} TP[k]}{\sum_{k=0}^{K-1} TP[k] + FN[k]} \tag{4.2}$$

$$F_1 = \frac{2pr \times re}{pr + re} \tag{4.3}$$

In (4.1 - 4.3), TP, FP, and FN represent true positive, false positive, and false negative binary descriptors indexed by prediction. These metrics are used on a frame-based level and a note-based level. We use the MIR community standard evaluation package *mir_eval* [57] to compute these scores for each experiment.

For frame-based evaluation, the frame-wise f_0 activity inferred from the notes predictions are compared against the ground truth frame-wise f_0 activation inferred from the ground-truth notes. The problem then becomes akin to that of MFE, where the goal is to detect active frequencies across frames using multi-class binary classification. The resolution of predictions is relatively arbitrary using this system, and does not necessarily correspond to the frame resolution of the model set by R, since it is reliant on the note predictions. However, as time resolution of original f_0 activity lessens, onset and offset times become more quantized, affecting this metric.

For note-based evaluation, there are several definitions for what constitutes a correct note, each adding a further layer of complexity to the task. The most simple definition of a correct note prediction is one where the estimated f_0 is within half a semitone interval of the true value, and there is a corresponding onset time estimation within 50 ms of the true value. Another more challenging definition builds upon the previous, but also requires that there exist a corresponding offset time estimation within the larger of either 50 ms or 20% of the ground truth duration. For piano-based experiments, both datasets include the MIDI-encoded strike velocity for each note annotation, allowing us to add a valid velocity estimation to all the previous. The metric which factors velocity is described in [7].

4.3 Training Procedure

In [7, 1], a batch size of 8 was used with samples of sequence length 20 seconds. Due to the increased computational demands wrought by learning a filterbank, it was necessary to reduce both of these parameters. In experiments 1 through 5, the batch size was set to 4 and the sample length to 10 seconds. In experiments 6 and 7, these parameters were 3 and 20, respectively. No precautions were made to avoid choosing audio sequences which began or ended during a note activation. Due to timing constraints, we were unable to train on MAESTRO for 670k steps as detailed in [1], needing instead to stop at 200k steps. A coarse manual search was conducted in order to fix a separate learning rate for the filterbank at $6E-6$. The learning rate of the transcription model was set to $6E-4$, and both learning rates decayed across

	Frame			Note			Note w/ off.			Note w/ off. & vel.		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
FRW	67.68	35.82	45.51	84.43	43.56	56.90	35.01	18.22	23.72	29.83	15.74	20.40
FCQ	69.07	67.00	66.63	85.44	75.56	80.07	41.22	36.60	38.71	38.41	34.14	36.09
FVQ	68.79	62.40	64.07	85.08	72.36	78.04	40.40	34.47	37.12	37.88	32.37	34.84
TRW	69.42	50.71	57.10	85.81	60.15	70.30	40.02	28.25	32.92	35.17	25.03	29.08
TVQ	67.12	57.95	60.76	84.58	65.21	73.34	39.39	30.58	34.29	35.19	27.43	30.71
TRW2	67.41	59.19	61.60	85.34	65.62	73.96	39.90	30.84	34.68	35.86	27.81	31.23
TVQ2	66.08	61.07	62.06	84.19	66.75	74.26	39.55	31.46	34.95	36.07	28.79	31.93

Table 4.2: Results from evaluation using the acoustic recordings of the full-pieces contained within the *ENSTDkAm* and *ENSTDkCl* partitions of the MAPS dataset.

	Frame			Note			Note w/ off.			Note w/ off. & vel.		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
FRW	79.31	51.33	62.01	95.80	51.88	66.64	52.07	28.92	36.81	45.90	25.65	32.58
FCQ	93.31	82.22	87.34	98.86	87.90	92.99	80.83	72.02	76.11	78.95	70.40	74.38
FVQ	92.47	74.97	82.68	98.30	81.40	88.87	76.26	63.39	69.09	74.28	61.82	67.34
TRW	89.88	68.48	77.55	97.17	72.61	82.79	71.15	53.45	60.80	65.72	49.54	56.27
TVQ	91.13	74.45	81.82	97.79	78.94	87.15	74.41	60.31	66.47	70.45	57.22	63.00
TRW2	90.01	75.45	81.97	97.57	79.01	87.11	73.23	59.56	65.53	68.95	56.22	61.79
TVQ2	90.48	78.05	83.70	97.88	80.69	88.27	75.67	62.63	68.39	72.23	59.89	65.35

Table 4.3: Results from evaluation using the MAESTRO dataset testing partition.

10000 step intervals with a rate of 0.98 before being reset. As in [7], Adam optimizer [58] was used to learn the parameters of both the filterbank and the transcription model jointly.

4.4 Results

The results obtained using the real piano data from the MAPS dataset and the testing partition of the MAESTRO dataset are presented in Tables 4.2 and 4.3, respectively. We can see that for both datasets, the fixed CQT from FVQ outperformed all other fixed and fine-tuned filterbanks. In general, fine-tuning the VQT configuration did not improve performance. One exception is the frame score for TVQ2 when evaluated on MAESTRO, which is significantly higher than the fixed counterpart FVQ, though this is not reflected in the MAPS evaluation. In all other experiments, the fixed VQT was superior in performance for all other metrics. However, the margins are much smaller within the MAESTRO evaluation, which may suggest that learned

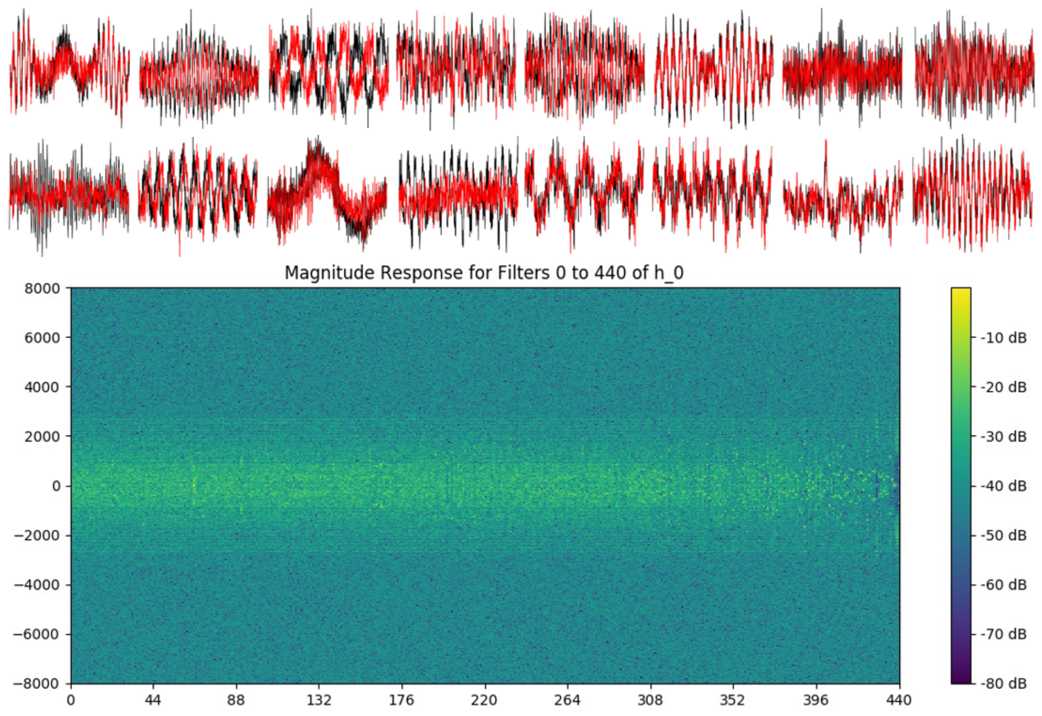

Figure 4.1: Subset of learned filters along with the overall frequency response for TRW. The filters are shown in no particular order, though the frequency response is ordered by spectral centroid.

filters generalize less effectively to other datasets with alternate recording conditions. Another interesting trend is the increase in performance in the second set of TRW and TVQ, where the batch size was lowered and the sequence length increased. Across most metrics, TRW2 actually outperforms TVQ, highlighting the importance of a longer sequence length during training and the efficacy of learning filters from random weights.

A small selection of the learned filters for each experiment where the filterbank weights were trained are presented in Figures 4.1 through 4.4. Quite possibly the most notable observation from the experiments is that the filterbanks which were trained from a random initialization were able to come very close to matching the performance of the fine-tuned VQT. VQT-like filters were learned in these experiments, though it is clear that the experiments with VQT initialization converge faster and more effectively since the filters are analytic from the start. This point is further

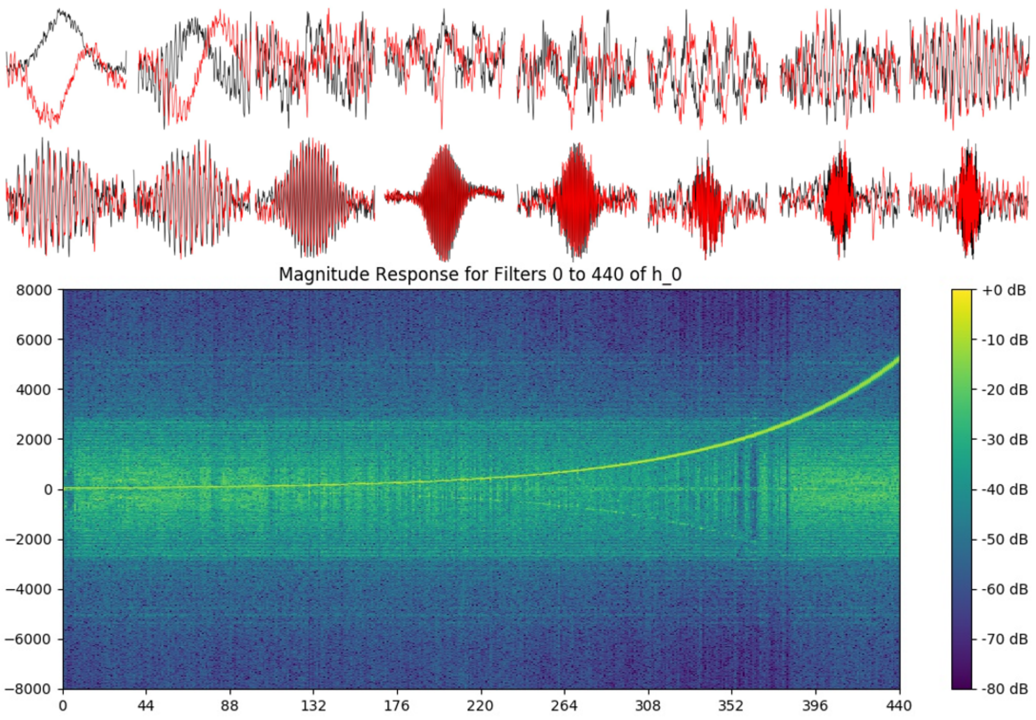

Figure 4.2: Subset of learned filters along with the overall frequency response for TVQ. The frequency response is ordered in accordance with an unmodified VQT.

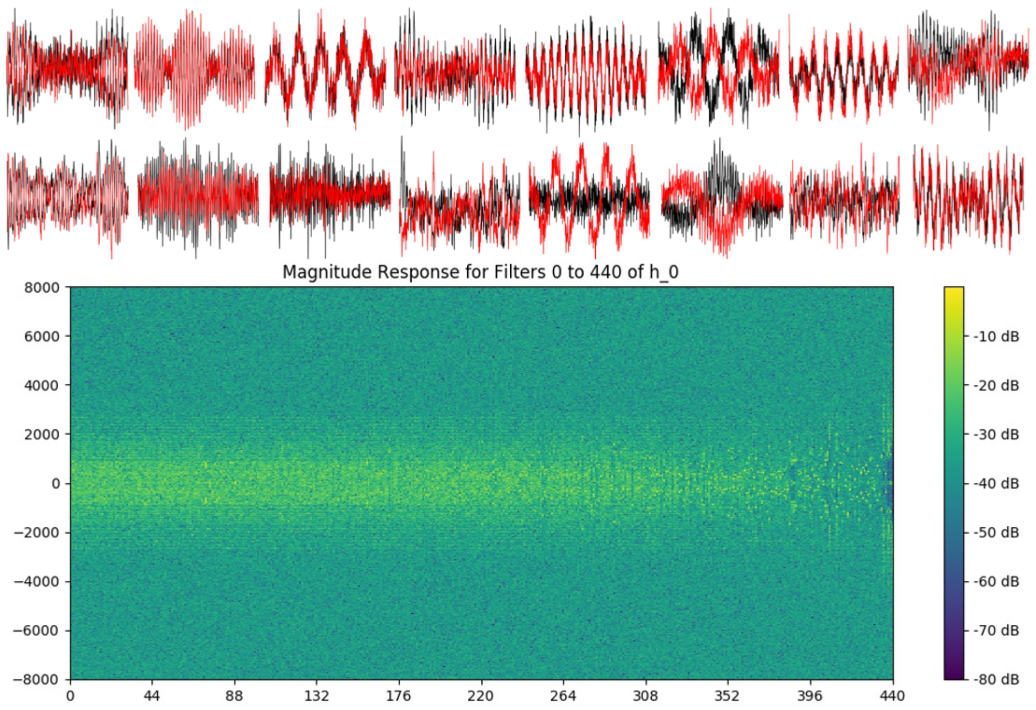

Figure 4.3: Subset of learned filters along with the overall frequency response for TRW2. The filters are shown in no particular order, though the frequency response is ordered by spectral centroid.

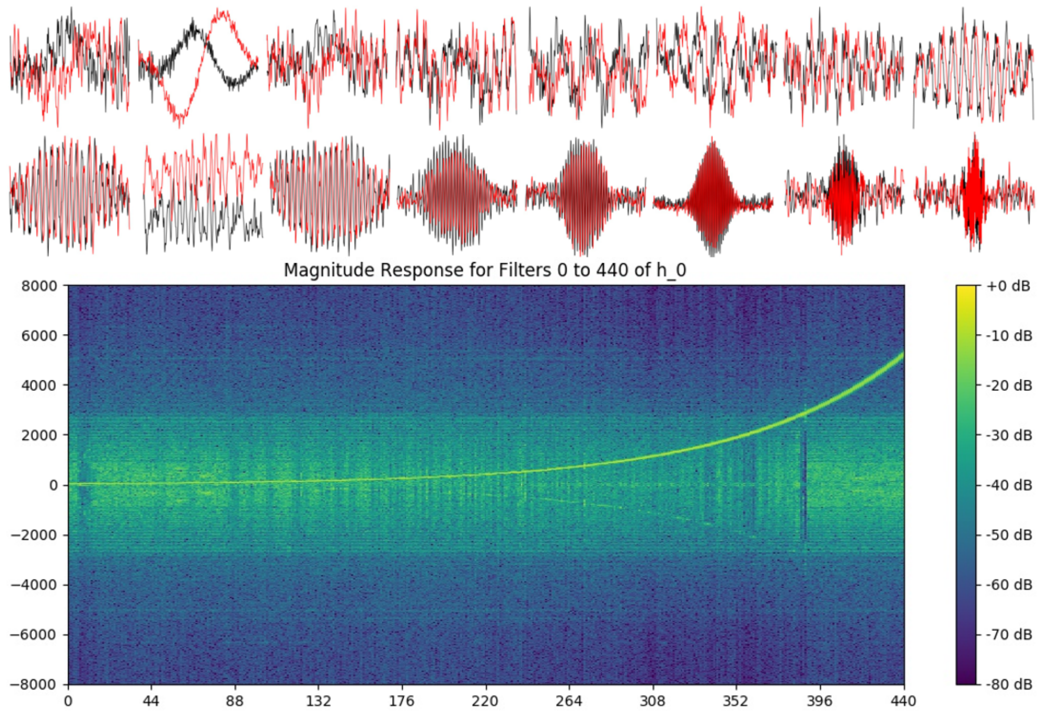

Figure 4.4: Subset of learned filters along with the overall frequency response for TVQ2. The frequency response is ordered in accordance with an unmodified VQT.

solidified by the fact that some filters from the random initialization remain noisy and unintelligible. In the random initialization, it appears that high frequency filters are rare, if learned at all.

The results and visualization for the experiments with fine-tuned VQT filterbanks suggest that in these situations, the optimization process simply added noise to the previously robust filters. The weights of most filters tend to resemble the initialization without the cleanliness of the standard counterparts, bolstering concerns laid out in [31]. The frequency responses for TVQ and TVQ2 further back this up, since they are asymmetric about zero and largely retain the strong response of the standard VQT with seemingly uniform noise across the response. It is difficult to say whether or not the learned filterbanks actually enhanced certain harmonics so that they could be prioritized at later stages, or if they actually modeled note events at specific frequencies. Answers to these uncertainties would require much deeper digging.

4.5 Future Work

The results suggest that the learnable filterbank struggles to find arrangements which are superior to the fixed TFR calculations. One reason for this may be the excess stochasticity that results from attempting to generalize from too few samples at a time during the training process. It also may be due to too little training, as is hinted by the amount of training that was needed to achieve state-of-the-art performance with the MAESTRO dataset [1]. One can only imagine that even more training than was previously necessary is required to learn something as delicate as a filterbank.

Several questions still surround the hyperparameter settings for the experiments. For instance, a coarse search was used to decide the learning rate, and it is not clear whether it was chosen optimally or not. In the future, a genetic algorithm may be a better way to choose the optimal setting dynamically. The batch size and sequence length clearly play a role in the experiments, and it seems that a longer sequence length is desirable, while still maintaining as large a batch size as possible. By making the weights of the filterbank learnable, there are limits to how much processing can be done at a time. As evidenced by the increase seen in experiments TRW2 and TVQ2, a longer sequence length and larger batch size may be necessary to efficiently learn filters which can generalize to a whole dataset.

It is not fully clear how the filterbanks should be constructed such that the best filters can be learned. The experiments show that the CQT is sufficiently more powerful than the VQT with γ set to 25, at least for piano transcription with the MAPS and MAESTRO dataset. Since γ can vary the transform from a CQT to a heavily saturated VQT, the subject of the experiments conducted in this work, it would be interesting to see if lowering γ would lead to a better filterbank layout, *i.e.* with enough space to learn good filters. The γ parameter chosen here allowed the filterbank to become sufficiently small so that it could be learned fast, giving it very little capacity compared to the CQT. The amount of bins per octave b was also not tuned. In the original release of the transcription model [7], the authors claimed that the CQT reduced performance with respect to the MFSCs calculation, though they did not mention what parameters they used; the MFSCs were calculated using 229 filters, resulting in a feature set roughly half the size of the 440 bins calculated in the transforms here. This may have assisted the model in convergence, since there was less information and less redundancy.

There was not enough time to perform experiments using 3D harmonic transforms, and it would be interesting to see if these can further improve transcription in the future. Once the hyperparameters of the base transforms and learning procedure are optimally tuned, the filterbank learning procedure may be able to enhance the response to certain harmonic frequencies without affecting the base transform for the first harmonic. However, there is no clear answer as to which harmonic transforms should be used in this representation, and each additional harmonic transform will require increased resources during training.

As learning from the waveform becomes more popular, it will be necessary to consider which architectural components are most important to filterbank learning, and how the model presented here may be further improved. Finally, since the learnable filterbank is extensible to other MIR tasks, it would be interesting to see if it can be used to improve other architectures, or assist in performing tasks where multiple instruments are present.

Chapter 5

Conclusion

Time-frequency calculations and features built upon them have long served as the primary input to any music classification system. While they are great for general-purpose music analysis, they are not tailored to any specific tasks. As such, in some cases it may be worthwhile to learn features directly from the waveform. In this work, an attempt was made to improve contemporary music transcription by jointly learning a filterbank with a transcription model. The filterbank can be initialized with random weights or it can be populated with the time-domain equivalent of the frequency response of a harmonic transform. Though there was no significant improvement to approaching the problem in this way, visualization of the learned filterbanks showed that they indeed are pulled toward modeling analytic filterbanks during training. In the future, it would be interesting to evaluate an increased amount of filterbank and architectural configurations, as well hyperparameter sets. For now, time-frequency calculations still reign supreme in the music space.

Bibliography

[1] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the maestro dataset," *arXiv preprint arXiv:1810.12247*, 2018.

[2] J. W. Kim, "Pytorch implementation of onsets and frames," https://github.com/jongwook/onsets-and-frames, 2019.

[3] E. J. Humphrey, J. P. Bello, and Y. LeCun, "Moving beyond feature design: Deep architectures and automatic feature learning in music informatics." in *ISMIR*, 2012, pp. 403–408.

[4] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, "Automatic music transcription: An overview," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2019.

[5] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri, "Automatic music transcription: challenges and future directions," *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 407–434, 2013.

[6] A. Klapuri, "Number theoretical means of resolving a mixture of several harmonic sounds," in *9th European Signal Processing Conference (EUSIPCO 1998)*, 1998, pp. 1–5.

[7] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, "Onsets and frames: Dual-objective piano transcription," *arXiv preprint arXiv:1710.11153*, 2017.

[8] M. Müller, *Fundamentals of music processing: Audio, analysis, algorithms, applications*. Springer, 2015.

[9] C. Weihs, D. Jannach, I. Vatolkin, and G. Rudolph, *Music data analysis: Foundations and applications*. Chapman and Hall/CRC, 2016.

[10] A. Klapuri and M. Davy, *Signal processing methods for music transcription*. Springer Science & Business Media, 2007.

[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] C. Schörkhuber and A. Klapuri, "Constant-q transform toolbox for music processing," in *7th Sound and Music Computing Conference, Barcelona, Spain*, 2010, pp. 3–64.

[13] G. A. Velasco, N. Holighaus, M. Dörfler, and T. Grill, "Constructing an invertible constant-q transform with non-stationary gabor frames," *Proceedings of DAFX11, Paris*, pp. 93–99, 2011.

[14] C. Schörkhuber, A. Klapuri, N. Holighaus, and M. Dörfler, “A matlab toolbox for efficient perfect reconstruction time-frequency transforms with log-frequency resolution,” in *Audio Engineering Society Conference: 53rd International Conference: Semantic Audio.* Audio Engineering Society, 2014.

[15] T. Necciari, P. Balazs, N. Holighaus, and P. L. Søndergaard, “The erblet transform: An auditory-based time-frequency representation with perfect reconstruction,” in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 498–502.

[16] E. Benetos, T. Weyde *et al.*, “An efficient temporally-constrained probabilistic model for multiple-instrument music transcription,” in *ISMIR*, 2015, pp. 701–707.

[17] R. Schramm, A. McLeod, M. Steedman, E. Benetos *et al.*, “Multi-pitch detection and voice assignment for a cappella recordings of multiple singers,” in *ISMIR*, 2017, pp. 552–559.

[18] A. McLeod, R. Schramm, M. Steedman, and E. Benetos, “Automatic transcription of polyphonic vocal music,” *Applied Sciences*, vol. 7, no. 12, p. 1285, 2017.

[19] A. Elowsson, “Polyphonic pitch tracking with deep layered learning,” *arXiv preprint arXiv:1804.02918*, 2018.

[20] F. Lins, M. Johann, E. Benetos, and R. Schramm, “Automatic transcription of diatonic harmonica recordings,” in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 256–260.

[21] R. M. Bittner, B. McFee, J. Salamon, P. Li, and J. P. Bello, “Deep salience representations for f0 estimation in polyphonic music.” in *ISMIR*, 2017, pp. 63–70.

[22] K. Choi, G. Fazekas, K. Cho, and M. Sandler, “A tutorial on deep learning for music information retrieval,” *arXiv preprint arXiv:1709.04396*, 2017.

[23] H. Purwins, B. Li, T. Virtanen, J. Schluter, S.-Y. Chang, and T. N. Sainath, “Deep learning for audio signal processing,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 2, pp. 206–219, 2019.

[24] Y. LeCun, “Learning invariant feature hierarchies,” in *European conference on computer vision.* Springer, 2012, pp. 496–505.

[25] P. A. Alvarado and D. Stowell, “Efficient learning of harmonic priors for pitch detection in polyphonic music,” *arXiv preprint arXiv:1705.07104*, 2017.

[26] S. Dieleman and B. Schrauwen, “End-to-end learning for music audio,” in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 6964–6968.

[27] P. Verma and R. W. Schafer, "Frequency estimation from waveforms using multi-layered neural networks." in *INTERSPEECH*, 2016, pp. 2165–2169.

[28] J. Lee, J. Park, K. L. Kim, and J. Nam, "Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms," *arXiv preprint arXiv:1703.01789*, 2017.

[29] J. Thickstun, Z. Harchaoui, and S. Kakade, "Learning features of music from scratch," *arXiv preprint arXiv:1611.09827*, 2016.

[30] J. Pons Puig, O. Nieto, M. Prockup, E. M. Schmidt, A. F. Ehmann, and X. Serra, "End-to-end learning for music audio tagging at scale," in *ISMIR*, 2018, pp. 637–644.

[31] J. Thickstun, Z. Harchaoui, D. P. Foster, and S. M. Kakade, "Invariances and data augmentation for supervised music transcription," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 2241–2245.

[32] T. N. Sainath, B. Kingsbury, A.-r. Mohamed, and B. Ramabhadran, "Learning filter banks within a deep neural network framework," in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, 2013, pp. 297–302.

[33] E. Cakir, E. C. Ozan, and T. Virtanen, "Filterbank learning for deep neural network based polyphonic sound event detection," in *2016 International Joint Conference on Neural Networks (IJCNN)*, 2016, pp. 3399–3406.

[34] D. Palaz, R. Collobert, and M. M. Doss, "Estimating phoneme class conditional probabilities from raw speech signal using convolutional neural networks," *arXiv preprint arXiv:1304.1018*, 2013.

[35] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals, "Learning the speech front-end with raw waveform cldnns," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[36] Z. Tüske, P. Golik, R. Schlüter, and H. Ney, "Acoustic modeling with deep neural networks using raw time signal for lvcsr," in *Fifteenth annual conference of the international speech communication association*, 2014.

[37] P. Ghahremani, V. Manohar, D. Povey, and S. Khudanpur, "Acoustic modelling from the signal domain using cnns." in *Interspeech*, 2016, pp. 3434–3438.

[38] N. Zeghidour, N. Usunier, G. Synnaeve, R. Collobert, and E. Dupoux, "End-to-end speech recognition from the raw waveform," *arXiv preprint arXiv:1806.07098*, 2018.

[39] Z. Zhu, J. H. Engel, and A. Hannun, "Learning multiscale features directly from waveforms," *arXiv preprint arXiv:1603.09509*, 2016.

[40] D. Palaz, R. Collobert *et al.*, "Analysis of cnn-based speech recognition system using raw speech as input," Idiap, Tech. Rep., 2015.

[41] Y. Hoshen, R. J. Weiss, and K. W. Wilson, "Speech acoustic modeling from raw multichannel waveforms," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 4624–4628.

[42] N. Zeghidour, N. Usunier, I. Kokkinos, T. Schaiz, G. Synnaeve, and E. Dupoux, "Learning filterbanks from raw speech for phone recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 5509–5513.

[43] H. Seki, K. Yamamoto, and S. Nakagawa, "A deep neural network integrated with filterbank learning for speech recognition," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 5480–5484.

[44] M. Ravanelli and Y. Bengio, "Speaker recognition from raw waveform with sincnet," in *2018 IEEE Spoken Language Technology Workshop (SLT)*, 2018, pp. 1021–1028.

[45] P. Taweewat and C. Wutiwiwatchai, "Musical pitch estimation using a supervised single hidden layer feed-forward neural network," *Expert Systems with Applications*, vol. 40, no. 2, pp. 575–589, 2013.

[46] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "Crepe: A convolutional representation for pitch estimation," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 161–165.

[47] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri, "Automatic music transcription: Breaking the glass ceiling," in *ISMIR*, 2012, pp. 379–384.

[48] S. Böck and M. Schedl, "Polyphonic piano note transcription with recurrent neural networks," in *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 2012, pp. 121–124.

[49] R. Kelz, M. Dorfer, F. Korzeniowski, S. Böck, A. Arzt, and G. Widmer, "On the potential of simple framewise approaches to piano transcription," *arXiv preprint arXiv:1612.05153*, 2016.

[50] S. Sigtia, E. Benetos, and S. Dixon, "An end-to-end neural network for polyphonic piano music transcription," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 5, pp. 927–939, 2016.

[51] C. Thomé and S. Ahlbäck, "Polyphonic pitch detection with convolutional recurrent neural networks," *Music Information Retrieval Evaluation eXchange (MIREX)*, 2017.

[52] R. Kelz, S. Böck, and G. Widmer, "Deep polyphonic adsr piano note transcription," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 246–250.

[53] B. McFee, M. McVicar, S. Balke, V. Lostanlen, C. Thom, C. Raffel, D. Lee, K. Lee, O. Nieto, F. Zalkow, and et al., "librosa/librosa: 0.6.3," http://doi.org/10.5281/zenodo.2564164, Feb 2019.

[54] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS Autodiff Workshop*, 2017.

[55] V. Emiya, R. Badeau, and B. David, "Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1643–1654, 2009.

[56] M. Bay, A. F. Ehmann, and J. S. Downie, "Evaluation of multiple-f0 estimation and tracking systems." in *ISMIR*, 2009, pp. 315–320.

[57] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. Ellis, and C. C. Raffel, "mir_eval: A transparent implementation of common mir metrics," in *ISMIR*, 2014, pp. 367–372.

[58] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.