NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

BSc THESIS

Incorporating Trainable Filterbanks in Deep Neural Networks for Music Transcription

Aikaterini-Maria A. Primenta

Supervisor: **Yannis Panagakis,** Associate Professor

ATHENS

APRIL 2024

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ενσωμάτωση Εκπαιδεύσιμης Συστοιχίας Φίλτρων σε Βαθιά Νευρωνικά Δίκτυα για Μουσική Μεταγραφή

Αικατερίνη-Μαρία Α. Πριμέντα

Επιβλέπων: **Γιάννης Παναγάκης,** Αναπληρωτής Καθηγητής

ΑΘΗΝΑ

ΑΠΡΙΛΙΟΣ 2024

BSc THESIS

Incorporating Trainable Filterbanks in Deep Neural Networks for Music Transcription

Aikaterini-Maria A. Primenta
S.N.: 1115201900160

SUPERVISOR: **Yannis Panagakis,** Associate Professor

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ενσωμάτωση Εκπαιδεύσιμης Συστοιχίας Φίλτρων σε Βαθιά Νευρωνικά Δίκτυα για Μουσική Μεταγραφή

Αικατερίνη-Μαρία Α. Πριμέντα

Α.Μ.: 1115201900160

ΕΠΙΒΛΕΠΩΝ: Γιάννης Παναγάκης, Αναπληρωτής Καθηγητής

ABSTRACT

In recent years, Automatic Music Transcription, the process of converting audio recordings into symbolic representations without the human intervention, has witnessed significant advancements and has been applied across various domains in the music field. Many existing approaches utilize Deep Neural Networks and rely on learning their input features directly from representations like log-mel spectrograms. This leads to challenges such as a high number of trainable parameters, limited adaptability and slow convergence. In this thesis, we tackle these challenges by proposing a new method to enhance piano transcription systems through the incorporation of trainable filterbanks for feature extraction. Drawing inspiration from SincNet, a Convolutional Neural Network architecture that implements parameterized sinc-based filterbanks, we aim to improve the accuracy and efficiency of an existing high-resolution piano transcription system. Our proposed framework achieves an Average Precision Score of 89%, which is comparable to but lower than that of the original method. However, it outperforms the original method in terms of the accuracy of onset and offset detections. The implementation of our proposed method is available at
`https://github.com/marikaitiprim/MusicTranscription-BScThesis`.

SUBJECT AREA: Music Transcription

KEYWORDS: Automatic Piano Transcription, Audio Signal Processing, Deep Neural Networks, Filterbanks, Log-Mel Spectrogram

ΠΕΡΙΛΗΨΗ

Τα τελευταία χρόνια, η Αυτόματη Μεταγραφή Μουσικής, η διαδικασία δηλαδή μετατροπής ηχογραφήσεων σε συμβολικές αναπαραστάσεις χωρίς ανθρώπινη παρέμβαση, έχει βιώσει σημαντικές προόδους και έχει εφαρμοστεί σε διάφορους τομείς της μουσικής. Πολλές υπάρχουσες προσεγγίσεις χρησιμοποιούν Βαθιά Νευρωνικά Δίκτυα και βασίζονται στην εκμάθηση των χαρακτηριστικών εισόδου απευθείας από αναπαραστάσεις όπως τα φασματογράμματα λογαριθμικής κλίμακας Mel. Αυτό οδηγεί σε προκλήσεις, όπως έναν υψηλό αριθμό εκπαιδεύσιμων παραμέτρων, περιορισμένη προσαρμοστικότητα και αργή σύγκλιση. Σε αυτήν τη διατριβή, αντιμετωπίζουμε αυτές τις προκλήσεις προτείνοντας μια νέα μέθοδο για τη βελτίωση των συστημάτων μεταγραφής πιάνου μέσω της ενσωμάτωσης εκπαιδεύσιμων φίλτρων για την εξαγωγή χαρακτηριστικών. Εμπνευσμένοι από το SincNet, μια αρχιτεκτονική με Συνελικτικά Νευρωνικά Δίκτυα που υλοποιεί παραμετρικά φίλτρα βασισμένα σε sinc συναρτήσεις, στοχεύουμε στην βελτίωση της ακρίβειας και της αποδοτικότητας ενός υπάρχοντος, υψηλής ανάλυσης, συστήματος μεταγραφής πιάνου. Το προτεινόμενο πλαίσιο επιτυγχάνει ένα Μέσο Ποσοστό Ακρίβειας 89%, το οποίο είναι συγκρίσιμο αλλά χαμηλότερο από αυτό της πρωτότυπης μεθόδου. Ωστόσο, συγκριτικά με την πρωτότυπη μέθοδο, αποδίδει καλύτερα στην ακρίβεια ανίχνευσης των ενάρξεων και απολήξεων των μουσικών νοτών. Η υλοποίηση της προτεινόμενης μας μεθόδου είναι διαθέσιμη στη διεύθυνση https://github.com/marikaitiprim/MusicTranscription-BScThesis.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Μεταγραφή μουσικής

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Αυτόματη Μεταγραφή πιάνου, Επεξεργασία Ηχητικού Σήματος, Βαθιά Νευρωνικά Δίκτυα, Φίλτρα, Φασματόγραμμα λογαριθμικής κλιμακας Mel

To my parents and my sister, Niki

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to Professor Yannis Panagakis for the invaluable support and guidance provided throughout the completion of this thesis. Furthermore, I wish to express heartfelt gratitude to my family and friends for their unwavering support and encouragement throughout my undergraduate studies, particularly to my dearest friend Kostas, whose inspiration led me to choose the topic of my thesis.

CONTENTS

1. INTRODUCTION **13**

2. RELATED WORK **15**

2.1 Early Methods in Automatic Music Transcription 15

2.2 Introduction of the Deep Learning in Automatic Music Transcription 15

2.3 Feature Extraction Methods 16

3. PROPOSED METHOD **17**

3.1 Baseline Method 17
3.1.1 Pre-Processing and Feature Extraction 17
3.1.1.1 Fourier Transform 17
3.1.1.2 Discrete Fourier Transform 17
3.1.1.3 Short-Time Fourier Transform 18
3.1.1.4 Spectrogram 19
3.1.1.5 Log-Mel Spectrogram 20
3.1.2 Piano Transcription System 20
3.1.2.1 Model Architecture 20
3.1.2.2 Inference 22

3.2 SincNet Architecture 23

3.3 Sinc-based Piano Transcription System 23

4. EXPERIMENTS **27**

4.1 Dataset 27

4.2 Training and Evaluation 27
4.2.1 Training from Scratch 27
4.2.2 Training with Pretrained Model 28

4.3 Results 30

5. CONCLUSION **32**

ABBREVIATIONS - ACRONYMS **33**

REFERENCES **36**

LIST OF FIGURES

3.1 High-resolution piano transcription system, adapted from the baseline method [1] 18
3.2 (a) Original waveform of the C major scale, (b) Spectrogram output of baseline method, (c) Spectrogram output using SincNet 25
3.3 (a) Original waveform of the C major scale, (b) Log-mel spectrogram output of baseline method, (c) Log-mel spectrogram output using SincNet 25
3.4 Log-mel spectrogram of the proposed method with min band parameter set to 0 26
3.5 Log-mel spectrogram of the proposed method with min band parameter set to 70 26

4.1 Results of our best trial on a 3 seconds part of ‘Waltz No. 2’ by Dmitri Shostakovich. (a) Music score notation, (b) Original waveform, (c) Log-mel Spectrogram of Baseline Method, (d) Log-mel Spectrogram of Proposed Method 31

LIST OF TABLES

4.1 Evaluation metrics after training both models for 1500 iterations 28
4.2 Evaluation metrics using the pretrained model 29

LIST OF ALGORITHMS

1 Inference for onset and offset times detection 22

1. INTRODUCTION

Music transcription is the task of converting acoustic music signals into various forms of music notation. Typically, acoustic music signals refer to audio recordings of music performances, while music notation takes the form of symbolic representations, such as sheet music or Musical Instrument Digital Interface (MIDI). This process has been a subject of fascination and endeavor for plenty of musicians and researchers, enabling a wide range of applications in various domains, such as music education, analysis and composition, acoustics and neuroscience. The motivation stems particularly from the desire to enhance musical studies and education, preserving and document the musical culture and enabling arrangements for different instruments or purposes. More specifically, transcribing music serves both professional musicians and researchers to analyze compositions and study the musical structures in detail, providing also valuable insights into the techniques, styles, and historical contexts. This way, music transcription can operate as a powerful educational tool for students who are willing to study music and develop their skills on a musical instrument. Musicians are also able to create arrangements and adaptations of existing music, such as transcribing music from one instrument or ensemble to another and creating more simplified versions or variations. In certain music genres, such as jazz, transcribing improvisations is essential for musicians to help them grow artistically and gain a richer understanding of this genre.

The difficulty of music transcription involves decoding all the subtle details of a musical performance, including pitch, timing, dynamics, and articulation, and transforming them into a widely recognized format. Capturing these details accurately demands a profound understanding of musical concepts and exceptional listening skills, given the complexity of the musical material, while it can also be time-consuming. Furthermore, technical limitations, such as poor audio quality or background noise, can further hinder the process.

Automatic Music Transcription (AMT) constitutes an improvement of manual music transcription methods, employing computational techniques to transcribe music without human intervention. According to an overview, introduced by Benetos *et al.* [2], there are four main categories in which AMT is divided: frame-level, note-level, stream-level and notation-level transcription. The progress in Artificial Intelligence in the early 21st century, impelled AMT to integrate machine learning models and techniques, without relying solely on the signal processing methods to extract musical information from audio recordings. Over the past decade, the development of deep learning in fields such as image and natural language processing has led research in AMT to take advantage of neural networks, demonstrating some astonishing results, although there is still progress to be made before achieving perfection.

This thesis focuses on the Automatic Piano Transcription (APT) using Convolutional Recurrent Neural Networks (CRNNs) and integrating filterbanks to enhance the training and improve performance. The piano serves as a focal point for research, as it is known for its ability to produce harmonically intricate sounds with varying velocities over time. The majority of the related works rely on extracting features using time-frequency signal representations, such as log-mel spectrograms [1, 3, 4, 5, 6, 7, 8, 9, 10, 11], without making them learnable and adaptable to the specific task. Therefore, we propose integrating a sinc-based approach with learnable filters from an architecture known as SincNet [12], into a simple yet remarkable piano transcription system [1]. Our motivation aligns closely with that of Thickstun *et al.* [13], focusing on reducing the number of

trainable parameters and optimizing feature extraction to achieve higher accuracy. The structure of our work unfolds as follows:

- The chapter 2 offers a brief overview of related work spanning the past two decades. It begins with an overview of early methods in Signal Processing and Machine Learning, followed by a discussion of more recent approaches based on Deep Neural Networks. A focus on feature extraction methods concludes this chapter.
- In chapter 3, we introduce our proposed method. We begin by presenting our baseline method providing essential technical background information on feature extraction. Then, we present the SincNet architecture and finally we propose our sinc-based piano transcription system.
- The chapter 4 concerns details about the dataset used for training and evaluation, followed by thorough explanations of all the experiments conducted. We present our results using relevant evaluation metrics and compare our proposed method with the original approach.
- In chapter 5, we extensively present the concluding remarks on the performance of our proposed method.

2. RELATED WORK

2.1 Early Methods in Automatic Music Transcription

In the first decade of the 21st century, research efforts in Automatic Music Transcription (AMT) primarily concentrated on frame-level transcription. This approach involves detecting both the number and pitch of musical notes present within a time frame. From traditional signal processing techniques [14] to modern machine learning methods, such as Support Vector Machines [15] and Bayesian approaches [16], Multiple Pitch Estimation (MPE), synonymous with frame-level transcription, marked a progress in the field. Additional methods have been introduced, notably Non-negative Matrix Factorization (NMF), which has proved to estimate effectively both spectral characteristics and temporal attributes of musical notes [17, 18].

Note-level transcription, also known as note tracking, represents a more contemporary and advanced task, gaining popularity in the 2010s. Pitch detection is performed for each time frame, with the detected pitches being linked over time to form complete notes. This method can be implemented in two principle ways: it can built upon the output of frame-level transcription, with techniques such as Hidden Markov Models (HMMs) [19] and spectral likelihood models [20] or it can function independently, such as estimating pitch, onset and offset in the same framework [5, 21, 22]. Some remarkable note tracking approaches include also unsupervised learning methods by constructing probabilistic models [23] and convolutional sparse coding methods [22, 24]. Finally, other research approaches have concentrated on notation-level transcription, which involves transcribing music audios into human-readable musical scores, usually by processing MIDI representations [25].

2.2 Introduction of the Deep Learning in Automatic Music Transcription

The breakthrough in deep learning led most researchers in AMT to utilize neural networks in order to enhance the performance of music transcribers and overcome obstacles derived from previous methods. While neural networks were regarded as distinct methods for achieving frame-level and note-level in the past, great advancements in AMT through deep learning have emerged in the past few years. Some of these methods depend on prior approaches, such as constructing high-level vocabularies in ABC notation [26], employing NMFs [11, 27] or utilizing Bayesian techniques [28]. Other approaches explore the exclusive use of complex Deep Neural Networks (DNNs) [29]. The research in AMT utilizing DNNs can be categorized into three main groups: (a) those leveraging Convolutional Neural Networks (CNNs) [4, 27, 30, 31], (b) those based on Recurrent Neural Networks (RNNs), especially Long Short-Term Memory networks (LSTMs) [5, 26] and (c) methods that blend both CNNs and RNNs, leading to Convolutional Recurrent Neural Networks (CRNNs) [1, 9, 32]. However, recent advancements have introduced transformer-based models [3, 6, 10, 33] to the forefront of AMT research. Additionally, some recent research attempts have focused on stream-level transcription, also referred to as Multi-Pitch Streaming (MPS) or timbre tracking, aiming at grouping pitches or notes into streams based on their timbre [34]. Regarding notation-level transcription, McLeod *et al.* [35] proposed integrating Music Language Models (MLMs) to MIDI post-processing, constructing the MIDI

Degradation Toolkit.

2.3 Feature Extraction Methods

Most of the recent research in AMT has focused on extracting spectro-temporal representations to feed their Neural Networks. A prevalent method involves utilizing the Short-Time Fourier Transform (STFT) followed by constructing a log-mel spectrogram representation [1, 3, 4, 5, 6, 7, 8, 9, 10, 11]. However, an emerging perspective suggests that the Constant-Q Transform (CQT) is more efficient in feature extraction due to its accurate representation of time-frequency information and lower dimensionality [30, 32]. Another approach aims at reconstructing both representations to improve accuracy [36]. Although Mel-Frequency Cepstral Coefficients (MFCCs) have been less prevalent in recent AMT research compared to other feature representations, they have been utilized in some studies, such as in the work by Simonetta *et al.* [27].

While these feature extraction methods are quite popular in AMT, they are not always considered as the most suitable choices, due to their high number of learnable elements and limited adaptability to specific tasks. Some research approaches in speech-related tasks, such as speech recognition, have proposed replacing fixed filterbanks with learnable ones [12, 37], or integrating learnable filterbank layers to boost the output of STFT [38]. In the music-related tasks, there have been fewer suggestions tackling this issue. Thickstun *et al.* [31] suggested learning features from scratch, introducing learnable window functions, while a more recent work put forward a fully adaptable system, known as LEAF [39], which aims at to replacing mel-filterbanks and reducing the trainable parameters.

3. PROPOSED METHOD

3.1 Baseline Method

Our proposed method is based on the work of Kong *et al.*[1], who introduced a high-resolution piano transcription system to detect piano notes' and pedal's onset and offset times. Our work focuses on the architecture of note transcription, as shown in Figure 3.1. The whole system is implemented using Python and PyTorch deep learning toolkit [40]. We present briefly this work in the following sections.

3.1.1 Pre-Processing and Feature Extraction

The first step involves extracting features from the data and converting them into log-mel spectrogram representations, which serve as inputs to the system. In order to do so, preprocessing of the data is required. The data comprises stereo audio recordings, which are first converted into mono and resampled to 16 kHz. The choice of a cutoff frequency of 16 kHz is appropriate as it covers the frequency range of the highest note, C8, on a piano, which is 4186 Hz. Then, the audio recordings are divided into 10-second clips and the feature extraction using log-mel spectrograms can begin. Let us provide some technical background to explain the construction of log-mel spectrograms.

3.1.1.1 Fourier Transform

The principle way to extract features from audio recordings is by using Fourier Transform. The Fourier transform (FT) is an integral transform that converts a function into a form that describes its frequencies. Frequency plays a significant role particularly in audio analysis, as it is the main characteristic that defines the pitch, the timbre, and the texture of a sound wave. Thus, the frequency components of a signal are extracted by FT. To be more precise, for each frequency $\omega \in R$, the Fourier transform produces a magnitude coefficient $d\omega$ and a phase $\phi\omega$ that explains to which extent the given signal matches a sinusoidal ($\sin$) prototype oscillation of that frequency. One of the advantages is that the original signal can be reconstructed from the magnitude and phase coefficients with great accuracy [41]. In the context of music, these coefficients define the partials of each musical note, which represent the integer multiples of the fundamental frequency of a note. This frequency, known as pitch, serves as the lowest partial and is integral to defining the musical tone.

3.1.1.2 Discrete Fourier Transform

Now, we present an evolution of the continuous Fourier Transform, called Discrete Fourier Transform (DFT), which offers a more practical method for analyzing discrete-time signals in digital signal processing. Signals are sampled at discrete intervals nT, where T is the sampling period (respectively $Fs = 1/T$ is the sampling rate), in order to reduce the number of computations performed on the signal. Although important information might be lost through the sampling process, the sampling theorem guarantees that an original analog signal f can be reconstructed perfectly from its sampled version x, if f does not

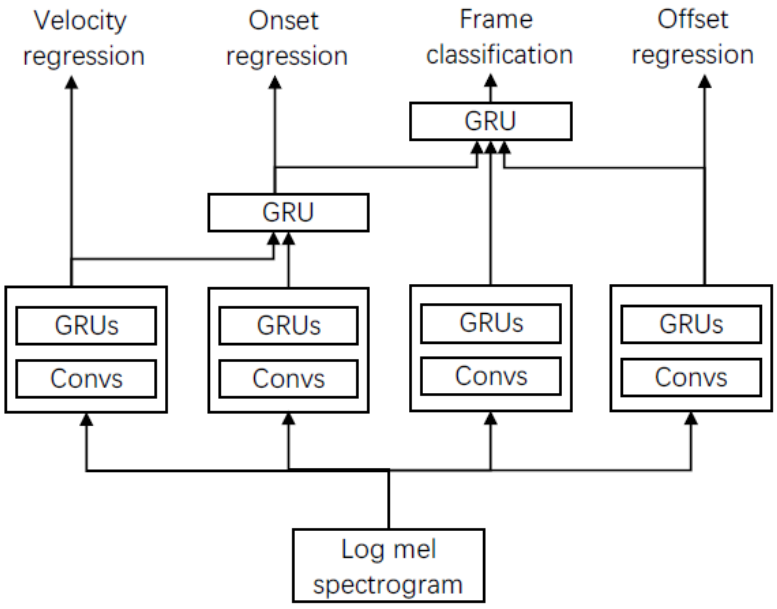

Figure 3.1: High-resolution piano transcription system, adapted from the baseline method [1]

contain any frequencies higher than $\Omega = Fs/2$, also known as the Nyquist frequency. In other case, the phenomenon of aliasing may occur. The general form for DFT is:

$$X(\omega) = \sum_{n \in Z} x(n) exp(-2i\omega n) \tag{3.1}$$

3.1.1.3 Short-Time Fourier Transform

The Fourier Transform, however, has a disadvantage; the magnitude provides frequency information that is averaged over the entire time domain (overall frequency content), hiding the precise timing these frequencies occur. This is a problem, as timing is crucial when analyzing musical pieces. Luckily, Short-Time Fourier Transform (STFT), introduced by D.Gabor in 1946, gave the solution. STFT considers a small section of the signal, called frame, and a window function, which is a non-zero function for only a short period of time and multiplies them to yield a windowed signal [41]. This way, frequency information can be obtained at different time instances by shifting the window function across time and computing a Fourier transform for each of the resulting windowed signals. The formula for the discrete STFT is:

$$X(m,k) = \sum_{n=0}^{N-1} x(n + mH) w(n) exp(-2ikn/N) \tag{3.2}$$

following a similar form to Equation 3.1 of DFT. The complex number $X(m,k)$, called magnitude, denotes the k^{th} Fourier coefficient for the m^{th} time frame and it is associated with the physical time position $T_{coef}(m) = m \cdot H/Fs$. The parameter H here represents the hop size, which is usually specified in samples and determines the step size in which the window will be shifted across the signal [41].

As mentioned previously, the STFT relies on a window function, whose length and shape

determine the performance of STFT. The parameter N for example denotes the length of the window but it also determines the duration of each frame (N/Fs seconds), as the frame size usually equals to the window size. The choice of the window function varies and depends on the application. However, the most commonly used windows are the hamming and the hanning window functions. It is worth noting that in most applications, the adjacent windows (or frames) overlap with each other in order to improve the accuracy of results. The overlap parameter is linked to both the hop size and the frame size as described by the equation:

$$frame\_size = overlap\_size + hop\_size \tag{3.3}$$

where all units here are measured in milliseconds.

Now, we can provide an accurate explanation of log-mel spectrogram by dissecting the term into two parts, log-mel and spectrogram.

3.1.1.4 Spectrogram

The Spectrogram is a two-dimensional representation of the squared magnitude of the STFT [41]:

$$Y(m,k) = |X(m,k)|^2 \tag{3.4}$$

This representation can be visualized as a color-coded image with two axes. The horizontal axis represents time and is determined by the frame indices m, while the vertical axis represents frequency and is determined by the frequency indices k. The squared magnitude symbolizes the energy (or power) of a signal present in each frequency-time bin within the Spectrogram. Furthermore, the real and imaginary parts enable Spectrograms to capture both magnitude and phase information to enhance more accurate results. In Spectrogram visualizations, the value $Y(m,k)$ is depicted by the intensity or color in the image at the coordinate (m, k) [41].

Our baseline method uses the TorchLibrosa toolkit [42] to extract the spectrogram for each audio clip. First, STFT is applied to extract the real and imaginary parts of the signal. The main parameters of STFT consist of a Hann window of size 2048 and a hop size equivalent to 160, determined by dividing the sample rate (16,000) with the number of frames (100) per second. This frame rate of 100 frames per second signifies that each frame has a duration of 10 ms.

Now, the spectrogram can be easily computed using equation 3.4. Given the hop size equivalent to 10 ms and a 10-second (10,000 ms) audio clip, the resulting spectrogram shape is 1001$\times$1025. Number 1001 denotes the number of frames in the clip, with the additional frame stemming from the padding with half windows on both sides of the audio clip's signal. Number 1025 signifies the number of frequency bins, calculated by the following formula:

$$freq\_bins = frame\_size//2 + 1 = window\_size//2 + 1 \tag{3.5}$$

In the present work, frame size and window size are identical, so we can easily calculate the number of frequency bins using the window size value. The formula involves dividing the window size by 2 to capture frequencies up to the Nyquist frequency ($f_s/2$). This approach ensures that all the information present in the original continuous signal is retrieved without introducing any aliasing artifacts. Frequencies beyond the Nyquist frequency would be reflected back and aliased into the lower frequencies, potentially causing distortion in the reconstructed signal.

3.1.1.5 Log-Mel Spectrogram

In many audio applications, the logarithm function is applied to the magnitude of the spectrogram, resulting in log-spectrogram, to emphasize lower energy components while retaining information about the higher ones. Furthermore, if the Mel scale is applied to the log-spectrogram, we obtain the log-mel spectrogram. The Mel scale is a perceptual scale of pitches that approximates the human auditory system's response to different frequencies. Its role in the spectrogram is to divide the frequency spectrum into distinct bins, where each one represents a perceptually distinct pitch.

In our baseline method, the number of mel bins is defined as 229 and the minimum and maximum cutoff frequencies are 30 Hz and 8000 Hz respectively. These values, together with the output spectrogram are finally used to extract the log-mel spectrogram, which has a shape of 1001×229.

3.1.2 Piano Transcription System

3.1.2.1 Model Architecture

After the feature extraction process, the features proceed through four submodules. Each submodule represents an acoustic model for velocity regression, onset regression, frame-wise classification and offset regression, as shown in Figure 3.1. All four models have identical architectures, which consist of convolutional blocks and bidirectional Gated Recurrent Unit (biGRU) layers, interspersed with batch normalizations, to normalize the outputs, Rectified Linear Units (ReLU), pooling, dropout and linear layers. Further explanation on the roles and applications of the layers and techniques within these four models is presented:

1. **Bidirectional Gated Recurrent Unit (biGRU)**: The Gated Recurrent Unit (GRU) is a variant of RNN designed to effectively capture sequential dependencies. GRU employs gating mechanisms, known as the reset gate and the update gate, to regulate the flow of information within the network at each time step. The reset gate determines the extent to which the previous hidden state is forgotten, while the update gate controls the incorporation of new input information into the hidden state. The output of the GRU is computed based on the updated hidden state, represented by the following equation:

$$h_t = (1-z_t)h_{t-1} + z_t h'_t \tag{3.6}$$

In this equation (3.6), z_t denotes the update gate and h'_t represents the candidate hidden state, which is determined by the reset gate and is responsible for selecting relevant portions of the previous hidden state h_{t-1}. The formula for the update gate is given by Equation 3.7:

$$z_t = \sigma(W_{hz}h_{t-1} + W_{xz}x_t + b_z) \tag{3.7}$$

where σ is the sigmoid function, W_{hz} and W_{xz} are learnable weight parameters, x_t denotes the input at the current time step and b_z is a bias parameter. The candidate hidden state h'_t is defined as:

$$h'_t = tanh(W_{hh}(r_t \circ h_{t-1}) + W_{xh}x_t + b_h) \tag{3.8}$$

where W_{hh} and W_{xh} are learnable weight parameters, r_t denotes the reset gate and b_h is a bias parameter. Finally, the formula for the reset gate is:

$$r_t = \sigma(W_{hr}h_{t-1} + W_{xr}x_t + b_r) \tag{3.9}$$

similarly to Equation 3.7.

The composition of two GRUs, one processing the input sequence in a forward direction and the other in a backward direction defines the bidirectional GRU (biGRU). This arrangement allows the model to capture information from both past and future contexts, enhancing its ability to understand sequential data.

2. **Rectified Linear Units (ReLU)**: ReLU is a non-linear activation function defined as:

$$R(z) = max(0, z) \tag{3.10}$$

meaning that for negative values of z, ReLU is set to 0, while for positive values of z, ReLU is set to z. Consequently, ReLU normalizes all negative values to zero.

3. **Pooling Layer**: Pooling is a downsampling operation used for reducing the spatial dimensions of feature maps while retaining important information. It involves applying a filter over each region of the feature map to obtain a single output value that summarizes the feature information. The main advantage of this technique is the reduction of the computational complexity and the number of learnable parameters. In our case, pooling is performed along the frequency axis to reduce sizes, preserving the temporal resolution of the transcription.

4. **Dropout**: Dropout is a method employed in neural networks to selectively deactivate nodes, spanning both the input and hidden layers. During training, a set of nodes, determined by a dropout probability p, is randomly chosen to be removed. Consequently, this action leads to the creation of a modified network structure, with the connections of the deactivated nodes being temporarily eliminated. Dropout prevents the network from relying excessively on specific nodes or features, promoting robustness and preventing overfitting.

The outputs of the four models are handled as follows. First, the velocity and onset regression outputs are concatenated and utilized as input for another biGRU layer. Then, this concatenated output along with the offset regression and the frame-wise classification outputs form the input for a final biGRU layer. The total loss function l_{note} to train this piano transcription system is the sum of all four submodules' loss functions described as follows:

$$l_{note} = l_{fr} + l_{on} + l_{off} + l_{vel} \tag{3.11}$$

where l_{fr}, l_{on}, l_{off} and l_{vel} are the loss functions of frame-wise classification, onset regression, offset regression and velocity regression respectively. The loss function for frame-wise classification is defined as:

$$l_{fr} = \sum_{t=1}^{T}\sum_{k=1}^{K} l_{bce}(I_{fr}(t,k), P_{fr}(t,k)) \tag{3.12}$$

where l_{bce} denotes the binary cross entropy, I_{fr} represents the frame-roll targets, P_{fr} represents the predictions and T and K are the number of frames and pitch classes accordingly. Pitch classes is equivalent to 88, which is the total number of piano notes. Similarly, the loss functions of onset and offset regression are:

$$l_{on} = \sum_{t=1}^{T} \sum_{k=1}^{K} l_{bce}(G_{on}(t,k), R_{on}(t,k)) \tag{3.13}$$

and

$$l_{off} = \sum_{t=1}^{T} \sum_{k=1}^{K} l_{bce}(G_{off}(t,k), R_{off}(t,k)) \tag{3.14}$$

G denotes the regression targets, while R represents the regression predictions. Finally, the loss function for velocity regression is:

$$l_{vel} = \sum_{t=1}^{T} \sum_{k=1}^{K} I_{on}(t,k) l_{bce}(I_{vel}(t,k), P_{vel}(t,k)) \tag{3.15}$$

where I_{on} and I_{vel} indicate the ground truth onsets and velocities respectively while P_{vel} denotes the predicted velocities.

3.1.2.2 Inference

Our baseline method includes an algorithm to process the outputs of the four submodules to high-resolution note events. Each note event is represented by a quadruple [piano note, onset time, offset time, velocity], which is calculated by Algorithm 1:

Algorithm 1 Inference for onset and offset times detection

Inputs: $R_{on}(t,k), R_{off}(t,k), P_{fr}(t), P_{vel}(t,k), \theta_{on}, \theta_{off}, \theta_{fr}$.
Outputs: Detected onset and offset times
for $k = 1, ..., K$ **do**
    for $t = 1, ..., T$ **do**
        if $R_{on}(t,k) > \theta_{on}$ and $R_{on}(t,k)$ is local maximum **then**
            Note onset of pitch k is detected.
            Calculate the velocity of the note by $P_{vel}(t,k) \times 128$.
        if ($R_{off}(t,k) > \theta_{off}$ and $R_{off}(t,k)$ is local maximum) or $P_{fr}(t) < \theta_{fr}$ **then**
            Note offset of pitch k is detected.
            Calculate the velocity of the note by $P_{vel}(t,k) \times 128$.

We provide a more detailed explanation of the algorithm. Once a frame has been detected to contain an onset, the exact onset time is determined by measuring its time distance from the frame center, utilizing the two adjacent frame centers. The accuracy of detection depends on whether the predicted onset regression is a local maximum and is greater than the threshold θ_{on}. Then, the onset velocity is derived by scaling the predicted velocity to a range of 0 to 127, used in the MIDI files. This scale denotes that higher integers correspond to louder notes. Similarly, for each detected onset, an offset is identified using

the same method, with the condition being either the frame prediction output lower than the frame threshold θ_{fr} or identical to the condition used for onset detection. Finally, all the onsets and offsets are paired to form piano notes.

3.2 SincNet Architecture

SincNet is a Convolutional Neural Network (CNN) architecture introduced by Ravanelli *et al.* [12] designed to improve performance in speaker recognition tasks. As indicated by its name, the SincNet architecture uses parameterized sinc ($sinc(x) = \sin(x)/x$) functions to implement band-pass filters for feature extraction. More specifically, SincNet performs time-domain convolutions with a predefined function, g, defined as:

$$g[n, f_1, f_2] = 2f_2 sinc(2\pi f_2 n) - 2f_1 sinc(2\pi f_1 n) \tag{3.16}$$

In the frequency domain, this function can be written as the difference between two low-pass rectangular filters:

$$G[f, f_1, f_2] = rect(\frac{f}{2f_2}) - rect(\frac{f}{2f_1}) \tag{3.17}$$

In these equations, f_1 and f_2 are the learnable parameters for low and high cutoff frequencies respectively. Indeed, the proposed band-pass filter is just an approximation of an ideal filter, thus windowing is essential. Ravanelli *et al.* proposed the use of Hamming window, although the results indicated no important difference in performance compared to Hann, Blackman and Kaiser window functions. Furthermore, owing to the symmetry of filters, computations can be conducted on only one side of the filter, accelerating the whole process.

The primary benefit of this technique is its focus on learning only two parameters, the low and the high cutoff frequencies of the filter, rather than learning all the filter elements. In this manner, regardless of the filter length, the number of learnable parameters remains constant, while the filters can be highly selective. Another advantage of this approach is its utilization of prior knowledge about the filter shape, while remaining adaptable to new data. This facilitates the learning of filter characteristics and accelerates convergence to the optimal solution. The overall analysis and results of SincNet proves that this architecture performs better in speaker recognition tasks and accomplishes all the advantages mentioned above.

3.3 Sinc-based Piano Transcription System

Our proposed method involves combining the SincNet architecture with the high-resolution piano transcription system to enhance feature extraction. Utilizing the learnable filters from the first layer of SincNet instead of learning all audio features from the STFT output, facilitates primarily the reduction of trainable parameters. To achieve this combination, we replace the spectrogram function from TorchLibrosa, as employed in the baseline work, with the Sinc-based convolution of SincNet. The SincNet convolution is structured to operate only with one (1) input channel (mono), which has been regulated in the preprocessing for our case and outputs the sinc filters activations. The main challenge lies in appropriately tuning the SincNet filters to align the output size

of the original method, by assigning the appropriate values to the arguments in the Sinc convolution. More precisely:

1. **Output channels:** The output channels argument denotes the number of output filters. In the original method, the corresponding value is the number of frequency bins, which is calculated by Equation 3.5. Thus, we assign the value 1025 to the output channels.

2. **Kernel size:** Kernel size represents the filter's length in samples. In Section 3.2, we discussed the importance of windowing the proposed band-pass filter demonstrating that the length of the filter corresponds to the window size. Therefore, we set kernel size to be equal to the window size (2048).

3. **Sample rate:** In Section 3.1.1, we presented the sample rate of the original method. The same sample rate will be used for our proposed method, which is 16 kHz.

4. **Stride:** Stride serves as a synonym for hop size, thus its value in the Sinc convolution must be identical to the original's one. As demonstrated in Section 3.1.1.4, the hop size is equal to 160.

5. **Padding:** The padding parameter controls the amount of zero-padding to be added to both sides of the input along the time axis before applying the convolution operation. This can be really useful in order to control the output size. By setting the padding parameter equivalent to the output channels (1025), we ensure that the output will have the same temporal dimension as the frequency bins (1001).

The Sinc convolution initializes filterbanks to be equally spaced in Mel scale, which is not considered in the computation of spectrogram in the TorchLibrosa toolikit. Thus, we opted to remove mel scaling to ensure comparability with the original spectrogram. Furthermore, the Sinc convolution actually replaces the STFT of TorchLibrosa in the original method. In order to generate the final spectrogram, Equation 3.4 must be applied to the sinc-based convolutional output. A notable difference in the output, yet, is that while STFT provides both real and imaginary parts of the signal, SincNet yields only the real part. In other words, SincNet computes the band-pass filters using only the magnitude response without explicitly considering the phase information. Therefore, we modify Equation 3.4 as:

$$Y(m,k) = |X(m,k)|^2 = |sinc\_output|^2 \tag{3.18}$$

Now, we can demonstrate the spectrogram representation using this proposed method. Figure 3.2 is divided into three parts, depicting a simple audio clip: the first part shows the waveform of the audio signal, while the second and the third show the original and the SincNet-based spectrograms, respectively. The audio recording contains all 8 piano notes of the C major scale, beginning from C3 note, whose fundamental frequency is approximately 130 Hz. Both spectrograms depict the frequency information of the notes played over time and exhibit significant similarities, including color consistency and precise time alignment. The evenly spaced lines observed over a defined duration represent the partials of each piano note. The frequency values of the partials range from 0 Hz to 8,000 Hz, corresponding to the audio signal's sample rate 16,000 Hz and the Nyquist frequency 8,000 Hz. It is evident that the higher partials tend to have darker colors, indicating lower energy levels and faster fading over time. The dB scale is utilized to provide a more convenient way to present the wide range of frequency values and their dynamic range.

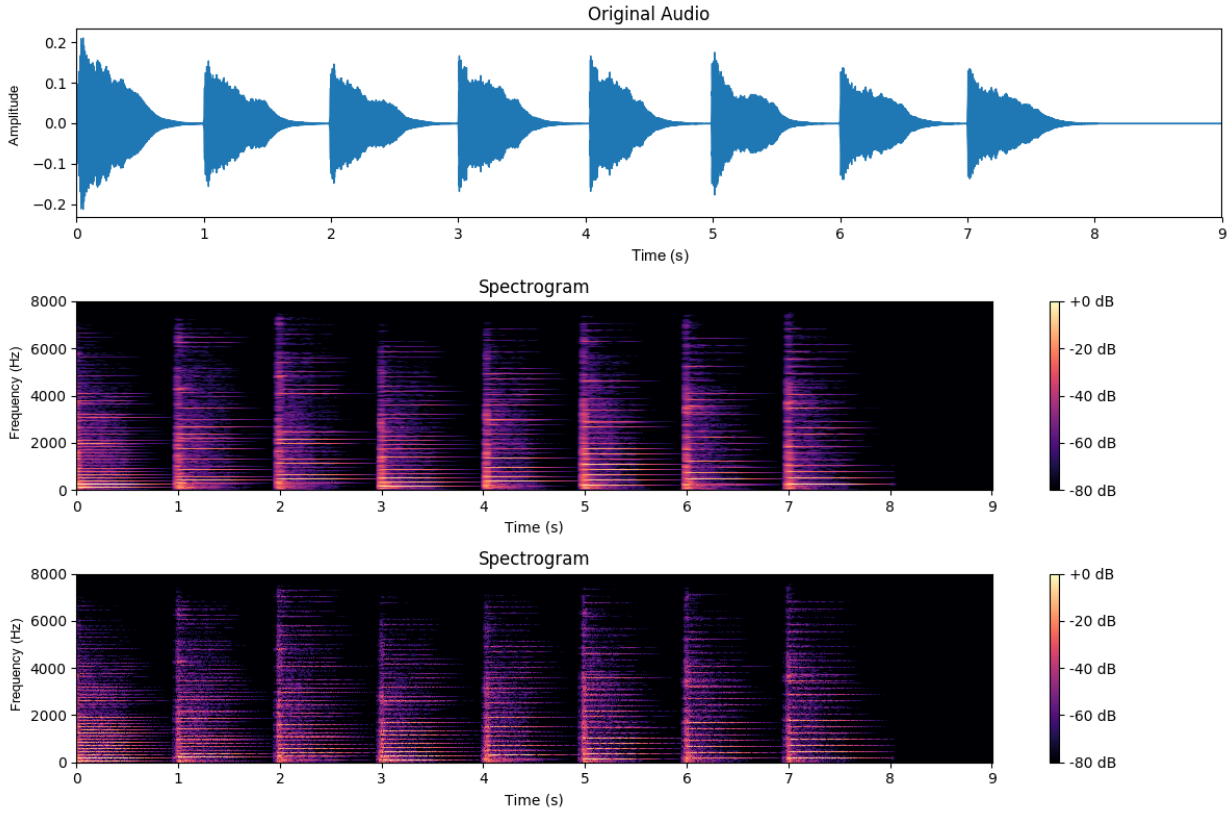

Figure 3.2: (a) Original waveform of the C major scale, (b) Spectrogram output of baseline method, (c) Spectrogram output using SincNet

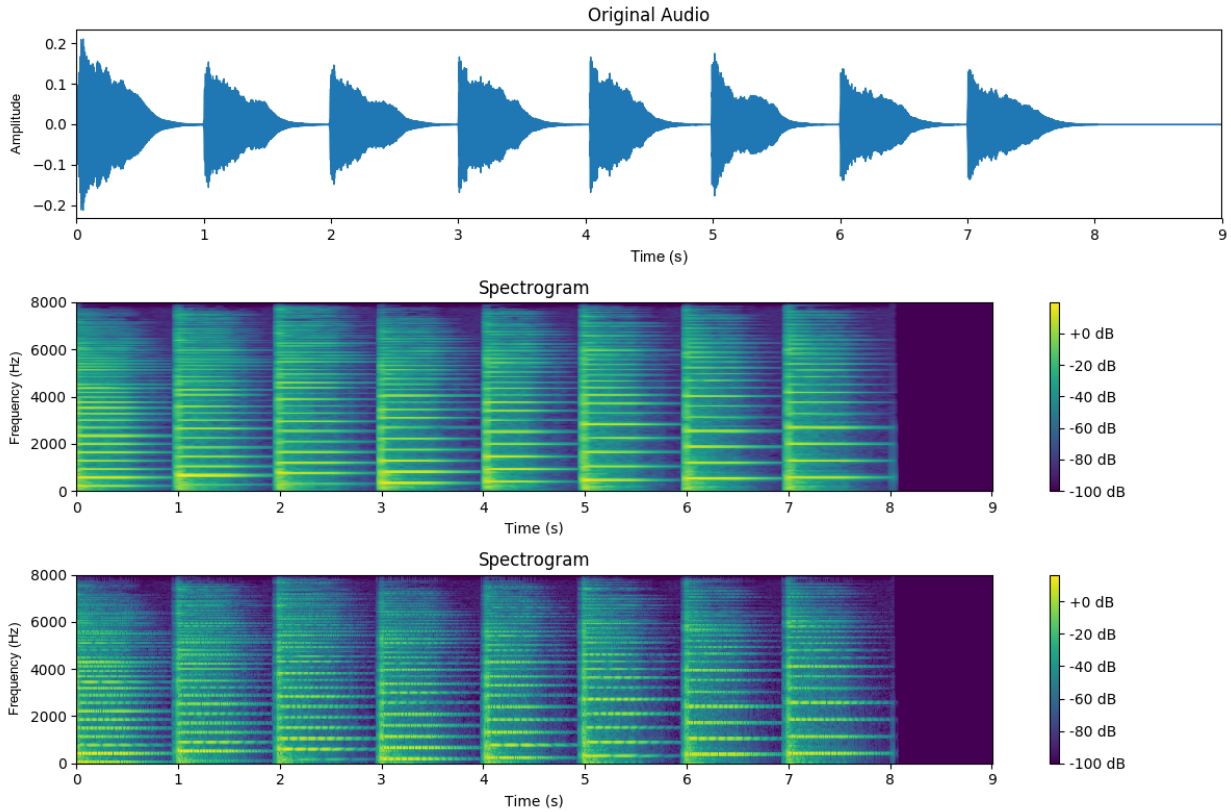

Figure 3.3: (a) Original waveform of the C major scale, (b) Log-mel spectrogram output of baseline method, (c) Log-mel spectrogram output using SincNet

Additionally, vertical lines are observed at the onset positions, corresponding to the noise-like transients that occur during the attack phase of the piano sound [41]. Another notable observation is that higher notes in the scale yield more distinct partials in the spectrogram representation. If we compare, for example, the first (C3) and the last (C4) piano notes, the partials in the latter are more widely spaced, indicating that higher fundamental frequencies result in fewer partials before reaching the Nyquist frequency. Similar and more prominent observations can be made after extracting the log-mel spectrogram using TorchLibrosa toolkit, as shown in Figure 3.3.

As elucidated in Section 3.2, SincNet architecture contains two trainable parameters which determine the filter learning process: the low and the high cutoff frequencies. As it is expected, given the low cutoff frequency, the bandwidth of a band-pass filter can be determined by the high cutoff frequency. Thus, in the SincNet implementation, the two learnable parameters used are the minimum low cutoff frequency and the minimum bandwidth frequency. Our proposed method is adapted to the requirements of the piano instrument, therefore the minimum low cutoff frequency parameter is configured to 21, aligning with the lowest frequency A0 on an acoustic grand piano. However, determining the optimal minimum bandwidth frequency is a more complex and requires further investigation. In the SincNet architecture, its default value is set to 50, which has proven effective in speech-related tasks. To explore its suitability for piano transcription, we conducted two experiments: one by adjusting to 0 and one to 70. The results are illustrated in Figures 3.4 and 3.5 accordingly, revealing differences in the thickness of the horizontal lines. More precisely, as the minimum bandwidth frequency decreases, the partials appear thinner (Figure 3.4) and as the minimum bandwidth frequency increases, the partials appear thicker (Figure 3.5). The explanation lies in the fact that a broader bandwidth allows more frequencies to pass through, leading to less selective filters. Consequently, the partials seem to encompass adjacent frequencies that do not align with their actual values. On the other hand, smaller values for the bandwidth size leads to highly selective filters that capture only a narrow range of frequencies (Figure 3.4). We decide to assign an intermediate value, 20 Hz, to mitigate the effects described. Figures 3.2 and 3.3 actually showcase the results of setting minimum band frequency to 20.

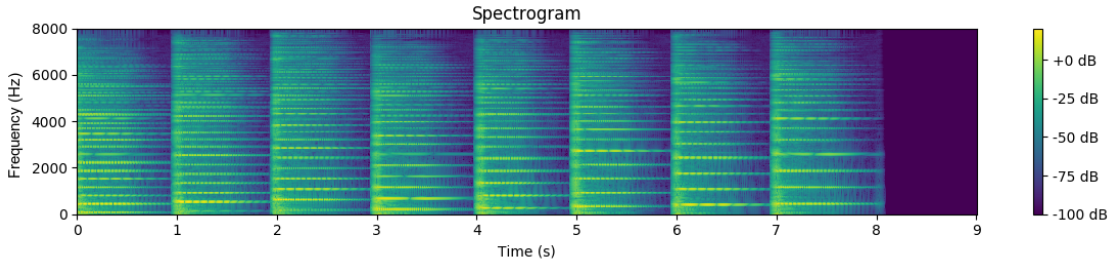

Figure 3.4: Log-mel spectrogram of the proposed method with min band parameter set to 0

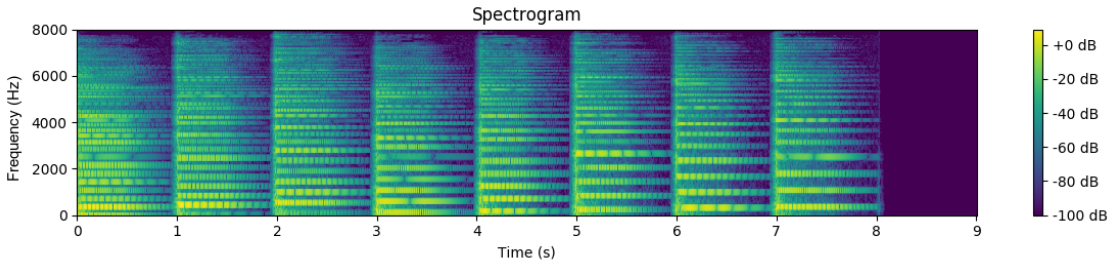

Figure 3.5: Log-mel spectrogram of the proposed method with min band parameter set to 70

4. EXPERIMENTS

4.1 Dataset

All the experiments were conducted using the MAESTRO dataset V2.0.0, introduced by Hawthorne *et al.* [43]. The same dataset was used by Kong *et al.* [1], our baseline method, to train and evaluate their proposed high-resolution piano transcription system. The MAESTRO dataset is a large dataset which contains over 200 hours of corresponding audio and MIDI recordings from ten years of International Piano-e-Competition. Yamaha acoustic grand pianos were utilized for the performances, facilitating the capture of high-precision MIDI data. Additionally, all audio and MIDI files were aligned with a time resolution of 3 ms approximately and contain meta-information about the composer, the title, and the year of the performance. Finally, the MAESTRO dataset is divided into 3 subsets: the training set, the validation set and the testing set. These subsets enable us to train and evaluate our proposed model effectively.

4.2 Training and Evaluation

To properly assess our proposed method and compare it with the original one, the training process is required. Similar to our baseline method [1], we utilize Adam [44] optimizer with a learning rate of 0.0005 for all of our experiments. However, due to constraints imposed by our Graphics Processing Unit (GPU), we adjust the batch size to either 10 or 11, as it cannot handle the batch size of 12 proposed by the original work. Our GPU is an NVIDIA GeForce RTX 4090 with CUDA version 12.2. The experimental procedure unfolds as presented in the following sections.

4.2.1 Training from Scratch

First, both models are trained from scratch to observe their performance over a limited number of iterations. We train the models for 1,500 iterations, evaluating them every 10 iterations, while decreasing the learning rate by a factor of 0.9 every 20 iterations. The results of the trials are presented in Table 4.1, where APS stands for Average Precision Score and MAE denotes the Mean Absolute Error. The Average Precision score is an evaluation metric that summarizes the precision-recall curve to a singular scalar, symbolizing the area beneath this curve. It ranges from 0 to 1, where higher scores correspond to high values in both precision and recall, while lower scores indicate low values in either of them across a range of confidence threshold values. The mathematical expression for APS is:

$$APS = \sum_{n} (R_n - R_{n-1}) P_n \tag{4.1}$$

In Equation 4.1, P and R are the precision and recall metrics, respectively, at the n^{th} threshold. Precision evaluates the model's ability to discern True Positives (TP) from all positive predictions (TP and False Positives (FP)) (4.2), while recall assesses the model's ability to identify True Positives (TP) among all actual positives (TP and False Negatives (FN)) (4.3).

Table 4.1: Evaluation metrics after training both models for 1500 iterations

	APS	MAE Onset	MAE Offset	MAE Velocity
Original method	Train: 11% Test: 11%	Train: 0.446 Test: 0.448	Train: 0.158 Test: 0.158	Train: 0.074 Test: 0.082
Proposed method	Train: 11% Test: **12%**	Train: **0.295** Test: **0.295**	Train: **0.074** Test: **0.076**	Train: **0.072** Test: **0.079**

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

In this context, the terms 'positive' and 'negative' refer to the piano notes under prediction. Conversely, the Mean Absolute Error represents the average absolute difference between predicted values and actual values and ranges between 0 and 1 too. Its mathematical formula is given by:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |true_i - predicted_i| \tag{4.4}$$

As demonstrated in Table 4.1, the MAE metric serves as the primary evaluation criterion for onset, offset, and velocity times. This selection is based on the characteristics of our baseline method, which employs a high-resolution piano transcription system generating continuous predictions for onset and offset times within the range of 0 to 1. Consequently, we are interested in the proximity of these predicted values to the ground truth. Similarly, although the velocity output comprises integer values instead of continuous ones ranging from 0 to 127, our evaluation objective remains consistent. Additionally, the APS serves as the pivotal metric for evaluating the classification system across the 88 classes, each representing a distinct piano note. For clarity reasons, we have decided to express the APS as a percentage in Table 4.1, while preserving the normalized scale in the values of MAE. The principal goal is the maximization of APS and the minimization of MAE. It is notable that both models exhibit relatively low APS values, proving the necessity of extensive training for thousands of iterations to reach satisfactory performance levels. Indeed, our baseline method required 200,000 iterations approximately to converge. However, from the current results, we observe that our proposed method demonstrates higher APS in the test set. Additionally, across both the train and test sets, the MAEs for onset, offset and velocity are much lower than the ones of the original method. This observation signifies a remarkable reduction in absolute error, aligning with the objective of error minimization.

4.2.2 Training with Pretrained Model

The results presented in the previous Section 4.2.1 prompt us to conduct further experiments regarding the training process. Fortunately, Kong *et al.* [1] have shared a

pretrained checkpoint file, containing the relevant weights derived from their experiments. Initializing the original model with these pretrained weights leads to outstanding results, as shown in the first row of Table 4.2. The baseline method achieves 95% and 91% APS for the train and test sets, respectively, which proves the efficiency of this high-resolution piano transcription system.

Encouraged by these outcomes, we proceed to integrate the pretrained model's weights into our proposed framework. Beginning with a few number of iterations (300), we reach an APS of 78% and 74% in the train and test sets, accordingly, as indicated in the second row of Table 4.2. This performance boost suggests that the pretrained weights offer a superior starting point compared to randomly initialized weights typically employed in training from scratch. It is also evident that the pretrained model has captured valuable features from the MAESTRO dataset from the previous training, leading to enhanced generalization capabilities.

Our following experiment involves training our model for 1,500 iterations. As demonstrated in the third row of Table 4.2, our proposed model reaches an APS of 80% in the training set and 77% in the test set, showing only a small improvement compared to the trial with 300 iterations. Notably, the MAEs exhibit a stagnation between the 300-iteration and 1500-iteration trials. This suggests that the performance boost obtained from the pretrained model's weights has reached its limit, prompting us to further train the model to confirm if any additional enhancements can be achieved.

In the original approach [1], training concludes after 200,000 iterations, with evaluations conducted every 5,000 iterations and the learning rate reduced by a factor of 0.9 every 10,000 iterations. To assess the performance of our proposed model, we opt to train it for 100,000 iterations, utilizing similar parameters to those in the original method. The results are summarized in the fourth row of Table 4.2. Our proposed model achieves an APS of 93% and 89% in the train and test sets, respectively. While the APS in our proposed method appears slightly lower than that of the original approach, the evaluation demonstrates that our model performs admirably well on the MAESTRO dataset. Notably, the onset, offset, and velocity MAEs in our proposed method closely mirror those of the original approach. Furthermore, we observe that the onset and offset MAEs are slightly lower in our proposed method, suggesting a minor enhancement in onset and offset detection.

Table 4.2: Evaluation metrics using the pretrained model

	APS	MAE Onset	MAE Offset	MAE Velocity
Original Method	Train: **95%** Test: **91%**	Train: 0.104 Test: 0.116	Train: 0.120 Test: 0.131	Train: **0.025** Test: **0.030**
Proposed Method (300 iterations)	Train: 78% Test: 74%	Train: 0.097 Test: 0.100	Train: 0.095 Test: 0.096	Train: 0.035 Test: 0.042
Proposed Method (1500 iterations)	Train: 80% Test: 77%	Train: 0.097 Test: 0.100	Train: 0.096 Test: 0.096	Train: 0.034 Test: 0.041
Proposed Method (100,000 iterations)	Train: 93% Test: 89%	Train: **0.100** Test: **0.115**	Train: **0.117** Test: **0.122**	Train: 0.027 Test: 0.033

4.3 Results

We claim that our most recent trial, which involves the training of our proposed model for 100,000 iterations, represents our optimal approach, yielding admirable APS scores and minimal MAEs. Despite attempting to extend the training duration to 200,000 iterations, matching the original method's recommendation, by reloading the weights from our best trial, we observed insignificant enhancements in the metrics. Figure 4.1 illustrates the outcomes of our most successful trial, evaluated on a 3-second piano clip from Dmitri Shostakovich's 'Waltz No. 2'. We deliberately selected a polyphonic piano transcription of a straightforward and well-structured composition, featuring a 3/4 time signature, to inspect the onset patterns within the spectrogram representation. The figure consists of four subplots: the first depicts the 3-second segment in standard Western music notation, incorporating staves, treble and bass clefs, accidentals, time signature, rests, and notes to convey pitch, rhythm, and tempo. The second subplot displays the original waveform of the audio clip, while the final two subplots present the log-mel spectrograms generated by both the original and proposed methods.

Our analysis of the two spectrograms reveals that the representation provided by our proposed method is notably clearer and more distinct in depicting the partials (horizontal lines) and onset positions (vertical lines). While both log-mel spectrograms exhibit considerable similarities, we observe that the onset positions are more accurately portrayed in our proposed representation at corresponding time instances. A notable example is the onset position just before the second second, where the vertical line representing the onset in the higher partials is more precise in our proposed representation compared to the original one. This onset aligns with the third measure of our audio clip, marking the beginning of notes in the treble clef.

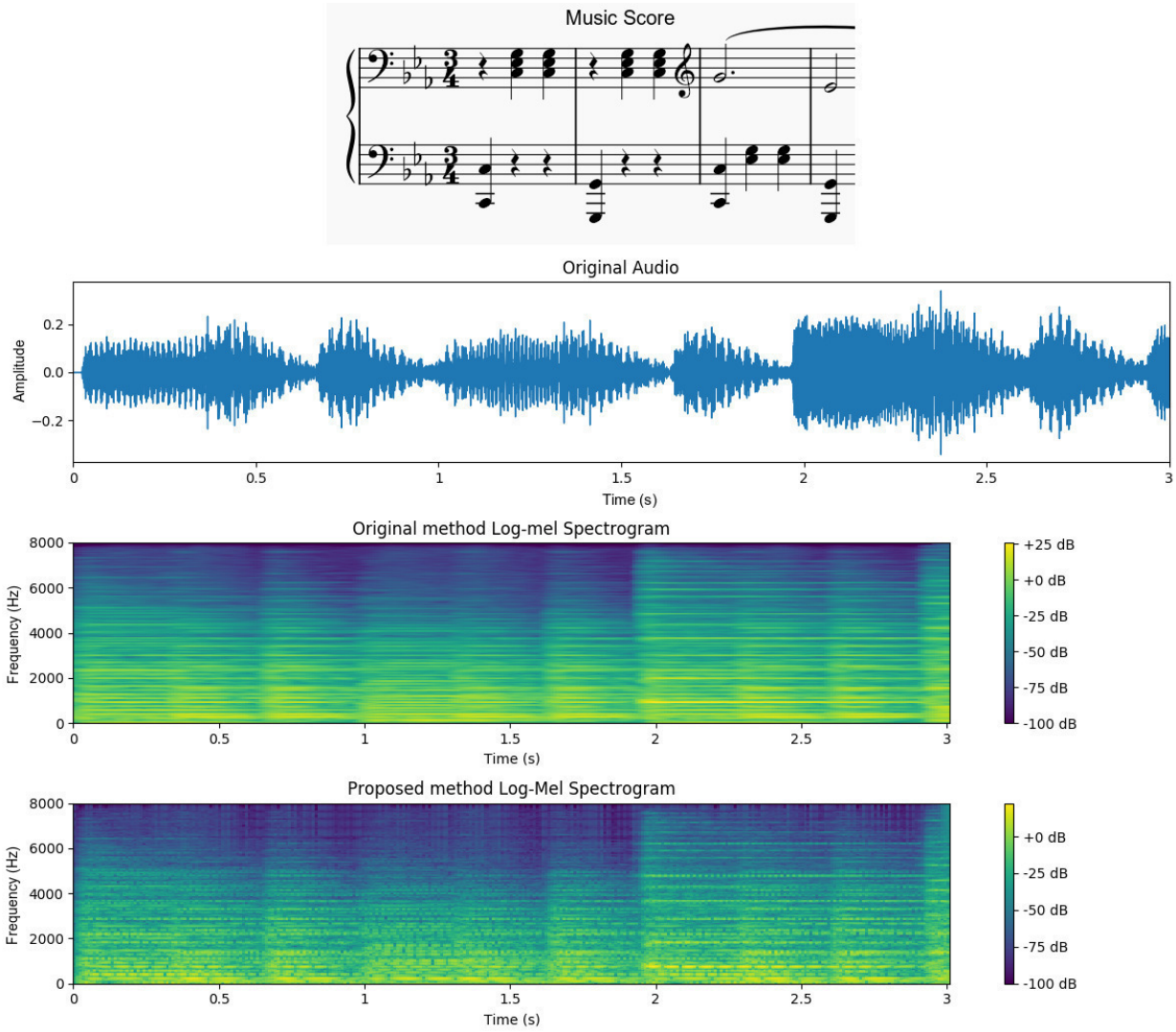

Figure 4.1: Results of our best trial on a 3 seconds part of 'Waltz No. 2' by Dmitri Shostakovich. (a) Music score notation, (b) Original waveform, (c) Log-mel Spectrogram of Baseline Method, (d) Log-mel Spectrogram of Proposed Method

5. CONCLUSION

In conclusion, we propose an enhancement to a piano transcription framework, by incorporating trainable filterbanks for feature extraction. Our approach is based on Kong *et al.*'s work [1], in which a high-resolution piano transcription system is implemented by using CRNNs and regressing the onset and offset times. Inspired by Ravanelli *et al.*'s work [12] in the speaker recognition tasks, we utilize the first layer of their proposed architecture, known as SincNet, as a replacement for the spectrogram computation in the original piano transcription framework. The results indicate that our proposed model achieves an 89% Average Precision Score in the test set, slightly lower than the 91% achieved by the original method. However, our approach showcases improved onset detection capability, by significantly reducing the number of trainable parameters. In the future, we aim to extend our model's applicability to different musical instruments, exploring the performance variations and we intend to experiment with modifications in the sinc-based learnable filterbanks, to refine the accuracy of our proposed model.

ABBREVIATIONS - ACRONYMS

MIDI	Musical Instrument Digital Interface
AMT	Automatic Music Transcription
APT	Automatic Piano Transcription
CRNN	Convolutional Recurrent Neural Network
MPE	Multiple Pitch Estimation
NMF	Non-Negative Matrix Factorization
HMM	Hidden Markov Model
DNN	Deep Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
MPS	Multi-Pitch Streaming
MLM	Music Language Model
STFT	Short-Time Fourier Transform
CQT	Constant-Q Transform
MFCC	Mel-Frequency Ceptral Coefficients
FT	Fourier Transform
DFT	Discrete Fourier Transform
GRU	Gated Recurrent Unit
biGRU	Bidirectional Gated Recurrent Unit
ReLU	Rectified Linear Unit
dB	Decibel
GPU	Graphics Processing Unit
APS	Average Precision Score
MAE	Mean Asolute Error
TP	True Positives
FP	False Positives
FN	False Negatives

BIBLIOGRAPHY

[1] Qiuqiang Kong, Bochen Li, Xuchen Song, Yuan Wan, and Yuxuan Wang. High-resolution piano transcription with pedals by regressing onset and offset times. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3707–3717, 2021.

[2] Emmanouil Benetos, Simon Dixon, Zhiyao Duan, and Sebastian Ewert. Automatic music transcription: An overview. *IEEE Signal Processing Magazine*, 36(1):20–30, 2019.

[3] Joshua Gardner, Ian Simon, Ethan Manilow, Curtis Hawthorne, and Jesse Engel. MT3: Multi-task multitrack music transcription. In *International Conference on Learning Representations*, 2022.

[4] Yoonchang Han, Jaehun Kim, and Kyogu Lee. Deep convolutional neural networks for predominant instrument recognition in polyphonic music. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(1):208–221, 2017.

[5] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse H. Engel, Sageev Oore, and Douglas Eck. Onsets and frames: Dual-objective piano transcription. In *Proceedings of the 19th International Society for Music InformationRetrieval Conference, ISMIR 2018, Paris, France, September 23-27,2018*, pages 50–57, 2018.

[6] Curtis Hawthorne, Ian Simon, Rigel Swavely, Ethan Manilow, and Jesse Engel. Sequence-to-sequence piano transcription with transformers. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

[7] Liwei Lin, Qiuqiang Kong, Junyan Jiang, and Gus G. Xia. A unified model for zero-shot music source separation, transcription and synthesis. In *International Society for Music Information Retrieval (ISMIR) Conference*, 2021.

[8] Ben Maman and Amit H Bermano. Unaligned supervision for automatic music transcription in the wild. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 14918–14934. PMLR, 17–23 Jul 2022.

[9] Miguel A. Román, Antonio Pertusa, and Jorge Calvo-Zaragoza. A holistic approach to polyphonic music transcription with neural networks. In *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019, Delft, The Netherlands, November 4-8, 2019*, 2019.

[10] Keisuke Toyama, Taketo Akama, Yukara Ikemiya, Yuhta Takida, Weimin Liao, and Yuki Mitsufuji. Automatic piano transcription with hierarchical frequency-time transformer. In *International Society for Music Information Retrieval (ISMIR) Conference*, 2023.

[11] Haoran Wu, Axel Marmoret, and Jérémy E Cohen. Semi-supervised convolutive nmf for automatic piano transcription. In *Proceedings of the 19th Sound and Music Computing Conference*, 2022.

[12] Mirco Ravanelli and Yoshua Bengio. Speaker recognition from raw waveform with sincnet. In *IEEE Spoken Language Technology Workshop (SLT)*, pages 1021–1028, 2018.

[13] John Thickstun, Zaid Harchaoui, Dean P. Foster, and Sham M. Kakade. Invariances and data augmentation for supervised music transcription. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2241–2245, 2018.

[14] Valentin Emiya, Roland Badeau, and Bertrand David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE Transactions on Audio, Speech, and Language Processing*, 18:1643 – 1654, 09 2010.

[15] Graham Poliner and Daniel Ellis. A discriminative model for polyphonic piano transcription. *EURASIP Journal on Advances in Signal Processing*, 2007, 01 2007.

[16] Paul Peeling, Ali Cemgil, and S.J. Godsill. Generative spectrogram factorization models for polyphonic piano transcription. *IEEE Transactions on Audio, Speech, and Language Processing*, 18:519 – 527, 04 2010.

[17] Paris Smaragdis and Judith Brown. Non-negative matrix factorization for polyphonic music transcription. In *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (IEEE Cat. No.03TH8684)*, pages 177–180, 2003.

[18] Emmanuel Vincent, Nancy Bertin, and Roland Badeau. Adaptive harmonic spectral decomposition for multiple pitch estimation. *IEEE Transactions on Audio Speech and Language Processing*, 18, 04 2010.

[19] Juhan Nam, Jiquan Ngiam, Honglak Lee, and Malcolm Slaney. A classification-based polyphonic piano transcription approach using learned feature representations. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2011.

[20] Zhiyao Duan and David Temperley. Note-level music transcription by maximum likelihood sampling. In *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, October 27-31, 2014*, pages 181–186, 2014.

[21] Tian Cheng, Matthias Mauch, Emmanouil Benetos, and Simon Dixon. An attack/decay model for piano transcription. In *International Society for Music Information Retrieval (ISMIR)*, 2016.

[22] Andrea Cogliati, Zhiyao Duan, and Brendt Wohlberg. Context-dependent piano music transcription with convolutional sparse coding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24:1–1, 12 2016.

[23] Taylor Berg-Kirkpatrick, Jacob Andreas, and Dan Klein. Unsupervised transcription of piano music. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[24] Andrea Cogliati, Zhiyao Duan, and Brendt Wohlberg. Piano music transcription with fast convolutional sparse coding. In *IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2015.

[25] Andrea Cogliati, David Temperley, and Zhiyao Duan. Transcribing human piano performances into music notation. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2016.

[26] Bob L. Sturm, João Felipe Santos, Oded Ben-Tal, and Iryna Korshunova. Music transcription modelling and composition using deep learning. *CoRR*, abs/1604.08723, 2016.

[27] Federico Simonetta, Stavros Ntalampiras, and Federico Avanzini. Acoustics-specific piano velocity estimation. In *IEEE 24th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–7, 2022.

[28] Ivan Nazarov and Evgeny Burnaev. Bayesian sparsification methods for deep complex-valued networks. In *37th International Conference on Machine Learning*, 2020.

[29] Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, Joao Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J Pal. Deep complex networks. In *International Conference on Learning Representations*, 2018.

[30] Rachel M. Bittner, Juan José Bosch, David Rubinstein, Gabriel Meseguer-Brocal, and Sebastian Ewert. A lightweight instrument-agnostic model for polyphonic note transcription and multipitch estimation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 781–785, 2022.

[31] John Thickstun, Zaid Harchaoui, and Sham Kakade. Learning features of music from scratch. In *International Conference on Learning Representations*, 2017.

[32] Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon. An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(5):927–939, 2016.

[33] Muqiao Yang, Martin Q. Ma, Dongyu Li, Yao-Hung Hubert Tsai, and Ruslan Salakhutdinov. Complex transformer: A framework for modeling complex-valued sequence. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4232–4236, 2020.

[34] Max Morrison, Caedon Hsieh, Nathan Pruyne, and Bryan Pardo. Cross-domain neural pitch and periodicity estimation. *ArXiv*, abs/2301.12258, 2023.

[35] Andrew McLeod, James Owers, and Kazuyoshi Yoshii. The midi degradation toolkit: Symbolic music augmentation and correction. In *Proceedings of the 21st International Society for Music Information Retrieval Conference*, pages 7. 846–852, 2020.

[36] Kin Wai Cheuk, Yin-Jvun Luo, Emmanouil Benetos, and Dorien Herremans. The effect of spectrogram reconstruction on automatic music transcription: An alternative approach to improve transcription accuracy. In *25th International Conference on Pattern Recognition (ICPR)*, pages 9091–9098, 2021.

[37] Tara N. Sainath, Brian Kingsbury, Abdel-rahman Mohamed, and Bhuvana Ramabhadran. Learning filter banks within a deep neural network framework. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 297–302, 2013.

[38] Quchen Fu, Zhongwei Teng, Jules White, Maria E. Powell, and Douglas C. Schmidt. Fastaudio: A learnable audio front-end for spoof speech detection. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3693–3697, 2022.

[39] Neil Zeghidour, Olivier Teboul, Félix de Chaumont Quitry, and Marco Tagliasacchi. Leaf: A learnable frontend for audio classification. In *International Conference on Learning Representations*, 2021.

[40] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, page 8024–8035. Curran Associates, Inc., 2019.

[41] Meinard Müller. *The Fourier Transform in a Nutshell*, pages 39–57. 08 2015.

[42] Qiuqiang Kong, Yin Cao, Turab Iqbal, Yuxuan Wang, Wenwu Wang, and Mark D. Plumbley. Panns: Large-scale pretrained audio neural networks for audio pattern recognition. *IEEE/ACM Transactions on Audio, Speech and Lang. Proc.*, 28:2880–2894, nov 2020.

[43] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *International Conference on Learning Representations*, 2019.

[44] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.