Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Polyphonic music transcription via deep neural networks

Bachelor Thesis

2020

Jozef Budáč

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Polyphonic music transcription via deep neural networks

Bachelor Thesis

Study Programme:	Computer Science
Field of Study:	Computer Science
Department:	Department of Computer Science
Supervisor:	prof. RNDr. Martin Škoviera PhD.
Consultant:	Mgr. Vladimír Macko

Bratislava, 2020
Jozef Budáč

54406334

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Jozef Budáč
Študijný program:	informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor:	informatika
Typ záverečnej práce:	bakalárska
Jazyk záverečnej práce:	anglický
Sekundárny jazyk:	slovenský

Názov: Polyphonic music transcription via deep neural networks
Prepis polyfonickej hudby pomocou hlbokých neurónových sietí

Anotácia: V bakalárskej práci sa zaoberáme problémom automatickej transkripcie polyfonickej hudby. Za použitia hlbokého učenia detekujeme výšky tónov z audia a následne porovnáme našu transkripciu s prislúchajúcim midi súborom pre daný zvukový záznam.

Cieľ: Cieľom práce je porovnať rôzne prístupy založené na metódach hlbokého učenia pre získanie transkripcie zo zvukového záznamu. Naše modely hodnotíme na niekoľkých uznávaných súboroch dát, ktoré obsahujú klasické hudobné nástroje, ako je klavír, a skúmame generalizáciu týchto metód do menej akademicky skúmanej oblasti typických slovenských folklórnych hudobných nástrojov.

Vedúci:	prof. RNDr. Martin Škoviera, PhD.
Konzultant:	Mgr. Vladimír Macko
Katedra:	FMFI.KI - Katedra informatiky
Vedúci katedry:	prof. RNDr. Martin Škoviera, PhD.

Spôsob sprístupnenia elektronickej verzie práce:
bez obmedzenia

Dátum zadania: 07.11.2019

Dátum schválenia: 25.11.2019 doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

....
študent vedúci práce

54406334

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Jozef Budáč
Study programme:	Computer Science (Single degree study, bachelor I. deg., full time form)
Field of Study:	Computer Science
Type of Thesis:	Bachelor´s thesis
Language of Thesis:	English
Secondary language:	Slovak

Title:	Polyphonic music transcription via deep neural networks
Annotation:	In this bachelor thesis, we deal with the problem of automatic polyphonic music transcription. With the help of deep learning, we detect pitches of the audio file and then compare our transcription with the corresponding midi file for the audio file.
Aim:	This bachelor thesis aims to compare different approaches based on deep neural networks to obtain transcription from an audio file. We evaluate our models on several acclaimed datasets that contain classical musical instruments such as piano and we explore the transfer of these methods to less academically explored domains of typical Slovak folklore musical instruments.

Supervisor:	prof. RNDr. Martin Škoviera, PhD.	
Consultant:	Mgr. Vladimír Macko	
Department:	FMFI.KI - Department of Computer Science	
Head of department:	prof. RNDr. Martin Škoviera, PhD.	
Assigned:	07.11.2019	
Approved:	25.11.2019	doc. RNDr. Daniel Olejár, PhD. Guarantor of Study Programme

....	....
Student	Supervisor

Acknowledgments: I would like to thank my supervisor prof. RNDr. Martin Škoviera PhD. for support and guidance. Furthermore, I would like to thank my consultant Mrg. Vladimír Macko, for his help with this work and his mentoring in this period of time. Without their help, I would not be able to properly finish this thesis. Special thanks go to Pavol Škoda, who helped me with creation of MIMAS dataset. Next, I want to thank Sigtia et al for their amazing research in automatic music transcription. Last, but not least, huge thanks goes to my family and friends.

Abstrakt

V tejto práci sa zaoberáme problémom automatickej transkripcie polyfonickej hudby pomocou hlbokých neurónových sieti, ktoré sú trénované predikovať výšky tónov ako v klasickej klavírnej hudbe tak aj v Slovenskej folklórnej hudbe. Ako zdroj inšpirácie sme si zobrali prácu publikovanú Sigtiom a replikujeme akustické modely navrhované v tejto práci. Vylepšujeme náš reprodukovaný Sigtiov ConvNet akustický model použitím populárnych konceptov súčasného strojového učenia, aby sme sa priblížili k jeho výsledkom. Náš prístup vedie k viac ako 9% relatívnemu zlepšeniu v našich experimentoch na súbore údajov MAPS. Ďalej sme vytvorili nový multi inštrumentálny midi zarovnaný súbor údajov zameraný na tradičnú slovenskú hudbu s cieľom vylepšiť výkon predikcie slovenských folklórnych multi inštrumentálnych záznamov.

Kľúčové slová: automatický prepis hudby, hlboké učenie, MAPS, MIDI

Abstract

We explore the field of automatic polyphonic music transcription by using a deep convolutional neural network which is trained to predict pitches of both classical piano music and Slovak folklore music. As a source of inspiration, we take an article published by Sigtia et al., and we replicate acoustic models used in their work. We improve our replicated Sigtia ConvNet acoustic model by using popular contemporary machine learning concepts in order to get closer to their results. Our approach results in over a 9% relative improvement in our experiments on the MAPS dataset. Furthermore, we create a novel Multi-instrument midi aligned sounds dataset focused on traditional Slovak music to improve the performance of predictions on Slovak folklore multi-instrumental records.

Keywords: automatic music transcription, deep learning, MAPS, MIDI

Contents

List of Figures

List of Tables

Introduction

Automatic polyphonic music transcription has not yet been satisfactorily solved. The existing approaches are mainly based on Non-Negative Matrix Factorization, or NMF-like methods using probabilistic latent component analysis (PLCA). In the work of Emiya et al. [4] they created the MIDI-Aligned Piano Sounds (MAPS) dataset which is an acclaimed benchmark for music transcription systems. The most relevant work to this thesis is Sigtia[23]. In this work, the researchers built the first AMT system using convolutional neural networks (among others) and outperforming the state-of-the-art approaches that used NMF.

Our goals are to replicate and improve acoustic models of Sigtia work and extend these models to predict both classical piano music and Slovak folklore songs obtained by typical folklore musical instruments. Since the Slovak folklore domain is less academically explored, we introduce Multi-instrument midi aligned sounds (MIMAS) dataset. We hope, this will motivate other researchers to explore the domain of Slovak folklore.

In the first chapter, we do a brief introduction to fundamental building blocks of music, music transcription and MIDI file format. In the second chapter, we present the commonly used methods in the field of audio processing. In the third chapter, we introduce the best performance architectures of Sigtia work and propose a potential improvement of one of them. In the fourth chapter, we present several acclaimed datasets include our own and evaluation of acoustic models on real piano pieces of music and Slovak folklore music.

Chapter 1

Foundamentals of music

Music is an art that has the power to make us happy, sad, motivated, more relaxed, concentrated, and so on. In other words, music is essential for us, and we can find it everywhere, even in nature. Many of us are only listeners, but there are people among us, who got a special gift from a God in the form of musical talent. These people are called musicians. They are playing or recording music from sheet music which has been written by some composer, or they are just improvising, and we like to listen to them. Musicians have to understand the fundamentals of music theory.

The music consists of regular patterns which characterize individual styles. Many songs follow these rules. For the purpose of this work, we will look at music as a sequence of tone combinations and will explore and imitate the patterns that they use.

1.1 Building blocks of music

Pitch and melody

The pitch represents how low or high the note is. We can order pitches on a frequency-related scale 1.1. Pitch is closely related to frequency, but they are not equivalent. Pitch is a property of sounds it describes a more psychological, each person's subjective perception of a sound wave. Frequency is an objective, scientific attribute measured in a units called hertz. Some musicians have a sense of absolute pitch, and they can quickly transcribe a piece of music. Standard pitch is the musical note **A** above middle C and is usually set at 440 Hz and serves as a general tuning standard for musical pitch. Not all musical instruments can produce notes with a clear pitch, and it can be unclear for music transcription systems.

A melody consists of pitches that must be musically meaningful. Melodies often contain notes from chords used in the song. The melody of traditional songs or folk songs often uses only the notes of a scale associated with the key of a given song. For example, if we take the national anthem of Slovakia "Nad Tatrou sa blýska" in **A** minor

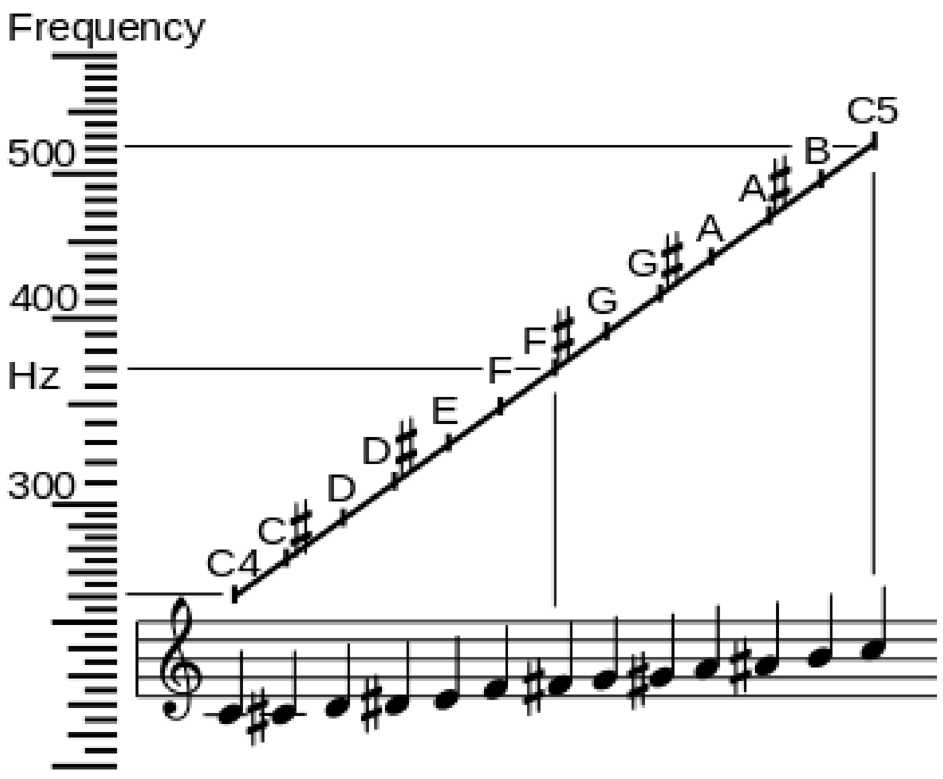


Figure 1.1: Frequency scale of notes[30].

scale, it contains pitches A, B, C, D, E, F and G.

Harmony and chords

Harmony refers to the part of music theory which studies the formation and relationships between chords. A chord is a combination of two or more simultaneous notes. There are regular patterns in chords, which describe how to chord a given song. These patterns are different for each music style, such as folk, jazz, and others. Chords also usually include notes from the melody theme used in the song. Chords are essential and powerful because they can turn the lovely melody to a truly bad song. Take a pure melody and accompany it by some jazz pattern chords and then some classical pattern chords. The result is two various songs although both have the same melody. So we must be careful when we chord songs because we can change its character.

Rhythm

Rhythm is a short, periodically repeated pattern in music. It is the basic temporal structure of music. When we are listening to music, there is a high probability of starting tapping our foot, or moving our head to the rhythm of the song. Beats per minute (in short bmp) is a term that represents the speed of the song. The beat is the fundamental unit of rhythm. For example, if you take a song with three beats per certain time intervals periodically repeated, this rhythm is known by the name waltz. If the song has four beats per time, in high probability it is the rhythm of tango. Metronome is a device which maintains stable tempo by producing audible clicks. This tempo can be usually set by a user. Musicians often use the metronome to practice songs. Notes are divided into groups by beats. Well-known and used are whole note(four beats), half note(two beats), quarter note(one beat), an eighth note(half of the beat), sixteenth note(a quarter of one beat) and thirty-second note(one-eighth of the beat).

Dynamics

In music terms dynamics or loudness are equivalent, but the term dynamics is more professional and preferable. Dynamics is split into several volume levels. Main two are piano and forte, which mean quiet and loud, respectively. Pianissimo, or too quiet, and fortissimo -too loud, are extremes. Moderate volume levels are represented by mezzo-piano and mezzo-forte. Changes from piano to forte is called crescendos, and the vice-versa changes are called decrescendos.

Tremolo or vibrato are rhythmic changes in loudness. They are troublesome for automatic transcription systems, since these systems may mistake peaks of the tremolo wave for a new note.

Loudness is an extremely significant factor in dealing with recordings of music. The quality of microphone, room acoustics or microphone placement may affect whether lower or higher pitches will be recorded at the same volume level, or not.

Timbre

Timbre is referred to as a tone colour or tone quality, and it is a core element in the recognition of sound sources. The harmonic content of a sound mainly determines timbre. Timbre allows distinguishing between various types of sound source, such as voices, string instruments, woodwind instruments, brass instruments, keyboard instruments and percussion instruments. Also for each class of musical instruments, timbre enables to identify distinct instruments, for example violin and cello, both string instruments or saxophone and clarinet, both in woodwind instruments.

1.2 Music transcription, MIDI and Sound Font

In music terminology, transcription is a process of converting acoustic music signals, (in many cases unannotated song, as, for example, an improvised solo) into some form of music notation. It is a complicated and challenging process because it consists of many subtasks, including pitch/multi-pitch detection, musical instrument recognition, voice separation, key detection, onset and offset estimation, dynamics, beat tracking and harmonic analysis. Therefore, music transcription or manual music transcription can be performed by someone only by someone with musical training, by listening to a piece of music repeatedly and rewriting the heard parts into sheet music.

Automatic music transcription

Automatic music transcription (AMT) is trying to come with a solution to automate the process of converting recorded audio into sheet music algorithmically. The first attempts towards the automatic music transcription were made in the 1970s by Moorer. We can see the illustration of music transcription in the picture 1.3.

This issue has not been satisfactorily solved yet, and it is a challenging problem, even for people, to deduce specific tones from a recording, especially when multiple instruments are playing at the same time. Spectrograms of different tones from a single instrument have common features that fluctuate in some way, depending on the pitch of the leading tone. Much of the source of the tone tells it to sound, so most digital instruments if they do not create a tone using a mathematical model, have a recorded tone beginning and a specified loop for arbitrary loop extension. We will use these methods to distinguish sound sources and determine their pitch. Speech recognition and image recognition techniques can be used as a source of inspiration, both are extremely popular topics in computer science research.

Musical Instrument Digital Interface (MIDI)

MIDI is a technical standard that describes a communications protocol, digital interface, and electrical connectors that connect a wide variety of electronic musical instruments, computers, and related audio devices for playing, editing and recording music [25].

MIDI transmits information such as note pitch, tempo, vibrato and many other elements necessary for music, but do not care information about the specific sound. MIDI files are not an audio signal, and they are much smaller than equal audio recorded audio files.

In this work, we operate with piano roll representation of MIDI files. As you can see on the picture 1.2, every piece of rectangles matches exactly one tone (y axe) in the time representation(x axe).This representation is academically called posteriogram. In other words, we are only interested in onset and offset times of each pitch. We do not include velocity or dynamics to our transcription system.

The way how to obtain MIDI files is to plug a MIDI keyboard, or musical instrument with MIDI output to some musical computer program or synthesizer and to record all the pressed keys., or we can use some virtual MIDI keyboard. This way we obtain pure midi files which can be reproduced by any type of musical instrument supported by our synthesizer. If we want to record real records with corresponding aligned MIDI file, we need to install very serious touch detectors to the music instrument, and this can be too complicated.

Sound Font

However, if the sound of the musical instrument, what we want is not in the database of our synthesizer, there exists one solution. We can create our own sound font of our musical instrument and add this sound font to synthesizer and use this font to reproduce the sounds from MIDI file.

Under the term sampling in connection with the creation of the musical font, we can understand a process of creation of samples from the sampled instrument, for the purpose of its synthesizing. One of the most used format is SoundFont (abbreviation sf2). It was created in the 90s, and it was pushed forward by the company Creative, together with its sound card Sound Blaster AWE32. Nowadays, the format SoundFont is overcome by other formats that enable further modification of the sound or scripting, however it is often used and is a good choice for many projects.

Database of such sound font consists of recordings of individual tones saved as wav file. It is important to record these files in the highest quality, since every mistake shows while playing the music. Furthermore, we need to be technically accurate in the technique while recording. Additionally, we can modify the properties of the sound

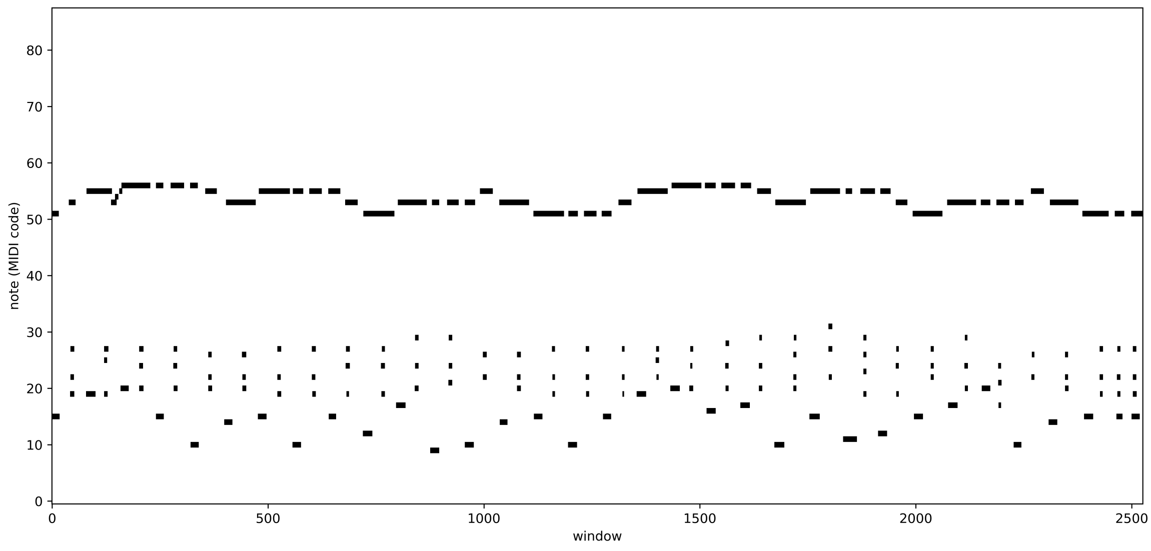


Figure 1.2: Piano roll representation of MIDI file of children Slovak song Kohutik jaraby.

such as an attack, decay, sustain, and release.

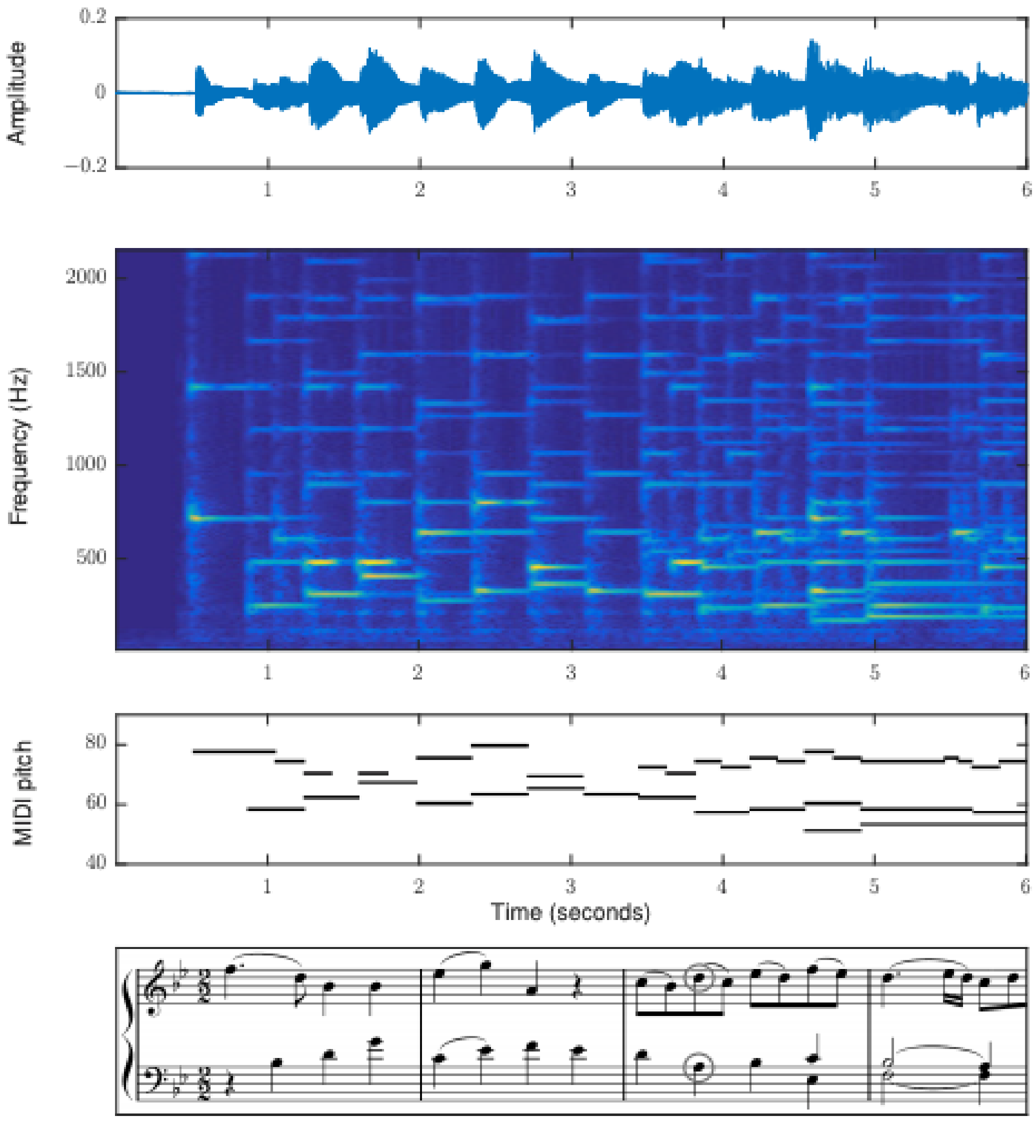


Figure 1.3: (a) Input waveform, (b) Internal time-frequency represantaion, (c) Output piano-roll represantaion (d) Output music score, with notes A and D marked in gray circles. The example corresponds to the first 6 seconds of W. A. Mozarts Piano Sonata No. 13, 3rd movement (taken from the MAPS database)[1].

Chapter 2

Audio analysis

An audio signal is a reproduction of the sound, usually can be represented by an electrical voltage or set of binary numbers for analogue signals or digital signals respectively. Digital audio systems represent audio signals in a variety of digital formats [9]. Sampling rate of audio is the number of samples of audio recorded per second. The sampling rate determines the maximum audio frequency that can be reproduced.

2.1 The Fourier Transform

In 1822, Joseph Fourier [2] pointed out that some functions could be represented as an infinite sum of harmonics, since then we know this as the Fourier Transform(FT). So, in other words, FT breaks down a domain signal into its fundamental frequencies. In practical situations, we generally have to deal with samples, real-valued discrete-time signals, denote $x(n)$, where n denotes discrete time.

$$DFT_x(k) = X(k) = \sum_{n=-\infty}^{\infty} x(n)e^{-j2\pi kn} \tag{2.1}$$

$$CFT_x(f) = X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}dt. \tag{2.2}$$

The continuous and discrete FTs map the signal from the time domain to the frequency domain; $X(f)$ and $X(k)$ are generally complex valued. The inverse Fourier transforms (IFTs) are also quite useful for music processing; they are defined below.

$$IDFT_X(n) = \sum_{k=-\infty}^{\infty} X(k)e^{j2\pi kn} = x(n) \tag{2.3}$$

$$ICFT_X(t) = \int_{-\infty}^{\infty} X(f)e^{j2\pi ft}df = x(t). \tag{2.4}$$

An efficient approach to computing DFTs is the $fast\ Fourier\ transform$ (FFT) algorithm [11].

2.2 Fast Fourier Transformation

While computing Discrete Fourier transform is a polynomial algorithm, meaning $O(n^2)$ time complexity for n data points, Fast Fourier Transformation reduces to $O(n\ log_2\ n)$ what is a linearithmic time. The main idea behind Fast Fourier Transformation algorithm is a method called divide-and-conquer.

On the surface, this might not seem like a big deal. Suppose that our n has large enough to make an enormous difference. For simplicity, let's say it took 1 nanosecond to perform one operation in CPU and take $n = 10^9$. Computation of DFT needs several decades, in particular 31.2 years. On the other side, FFT computes this efficiently in 30 seconds.

The basic idea is to break up a transform of n data points into two transforms of length $\frac{n}{2}$, and we do it repeatedly until we are left with groups of size 2, then we apply DFT. We devide it into even and odd indexed sub-sequences because they can be computed concurrently.

FFT is the purest and efficient way to split up a piece of music into its composite frequencies. However, a short-time FFT (in short STFT) is practically used for audio signal processing.

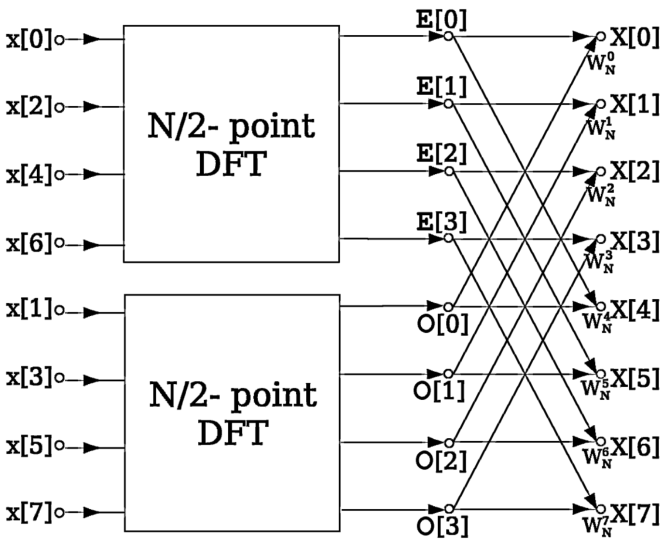


Figure 2.1: An example, FFT algorithm structure, using a decomposition into half-size FFTs [28].

2.3 Mel Frequency Cepstral Coefficients (MFFCs)

The mel frequency cepstral coefficients (MFFCs) of a signal are a small set of features (usually about 10-20) which concisely describe the overall shape of a spectral envelope. In Music Information Retrieval (MIR), it is often used to describe timbre or main feature in genre classification. Also, they are extremely popular in speech recognition systems.

Follow steps are required to obtain MFFCs.

1. Take the Fourier transform of a signal.
2. Map the powers of the spectrum obtained above onto the mel scale, using traingular overlapping windows.
3. Take the logs of the powers at each of the mel frequencies.
4. Take the discrete cosine transform of the list of mel log powers, as if it were a signal.
5. The MFFCCs are the amplitudes of the resulting spectrum.

2.4 Short-time Fourier Transformation

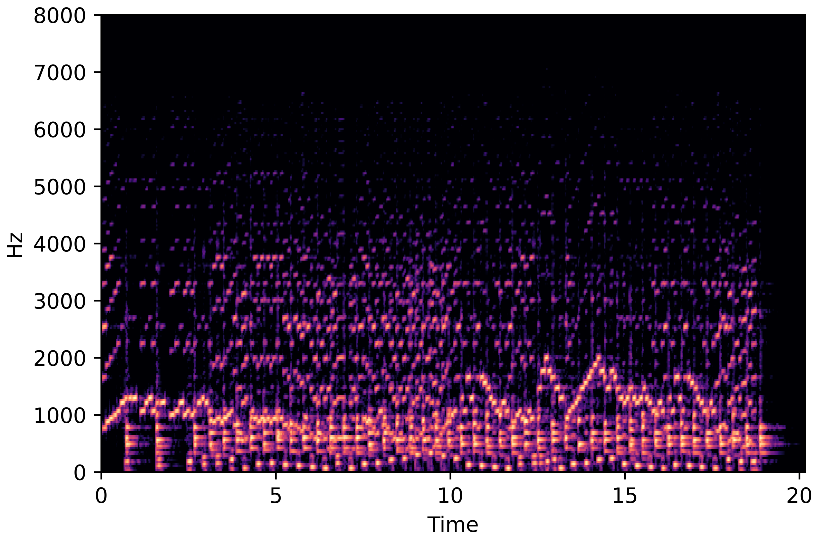
Short-time Fourier Transformation is a sequence of Fourier Transformations that decompose small sections of the signal called flowing window into component frequencies. Spectrogram is a visual representation of STFT, which represent the energy of frequency components at various times.

Choosing a size of window is an essential factor, but it depends on the case. For instance, in the case of speed recognition, the smaller pieces are necessary for getting more rapid changes in pitch. Beat detection algorithms also work with minimal frames.

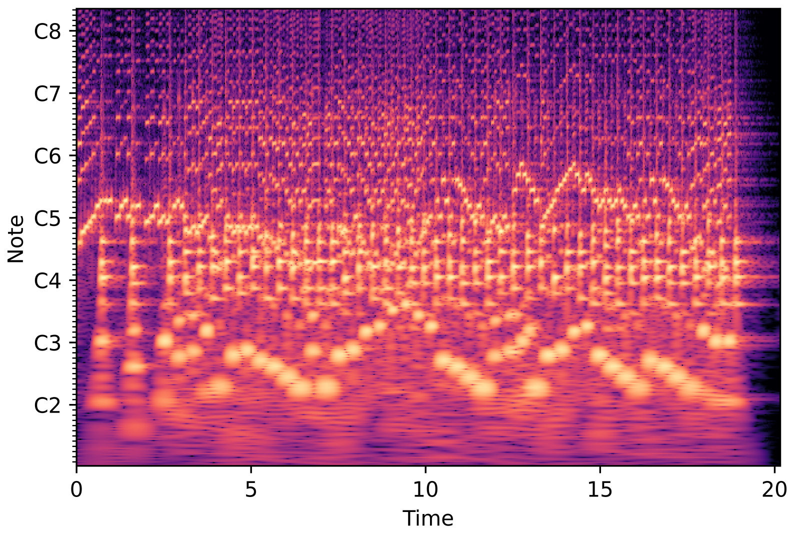
Consider an audio file sampled at 44.1 kHz in which rhythm is 120 beats per minute, then a whole note is 2000ms in length, a half note is 1000ms, a quarter note is 500ms, an 8th note is 250ms and 16th note 125ms. Now if we do a window size 4096 samples, it gives us frames in length 93ms, so it means it will capture even 16th note.

2.5 Constant Q-transform

An alternative to the STFT is the constant Q-transform. It transforms fragments of audio into spectrograms but in contrast with STFT uses logarithmically spaced filters to decompose the signal which makes these spectrograms more evenly spaced. It can be a relevant factor when we look at music as just a series of images, spectrograms.



(a) STFT



(b) constant Q-transform

Figure 2.2: Same audio processed by STFT and constant Q-transform.

Chapter 3

Machine Learning

The term Machine Learning was coined by Arthur Samuel in 1959 [20], an American pioneer in the field of computer gaming and artificial intelligence and stated that *"it gives computers the ability to learn without being explicitly programmed"*.

Machine learning algorithms build a mathematical model to make predictions or decision without being explicitly programmed to do so. Machine learning algorithms are split into two groups. The first one is called supervised and second one unsupervised. Supervised machine learning problems are further split into classification and regression problems and data is tagged with correct answers which we call labels. In this thesis we deal with a problem that does not have a clear programmatic solution but can be described by a lot of data examples. Hence supervised classification is the right approach to take.

3.1 Deep Learning

Deep learning is part of machine learning methods based on artificial neural networks. Depth allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in

- Automatic speech recognition,
- Natural language processing,
- Bioinformatics,
- Military,
- Image recognition,
- Medical Image Analysis,
- Image restoration,
- Visual art processing.

Deep learning discovers intricate structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep convolutional nets have brought about breakthroughs in processing images, video, speech and audio, whereas recurrent nets have shone light on sequential data such as text and speech [12].

3.1.1 Deep neural networks

Deep neural networks in short a DNN are kind of machine learning models that can be helpful for regression and classification problems. DNNs have one or more layers of non-linear transformations. The equation below shows an example of such non linear transformation.

$$h_{l+1} = f(W_l h_l + b_l). \tag{3.1}$$

Where W_l and b_l are the weight matrix and bias vector respectively for layer and f is some non-linear activation function of the units in the layer. To determine the parameters W_l nad b_l we make use of the Stoochastic Gradient Descent algorithm which is based on an idea of backpropagation. This process of finding the optimal parameters for DNN is called training. In case of AMT, the input to the DNN should be a frame of features, for instance, constant Q transform (CQT) and the DNN is trained to predict pitches in the frame at a particular time. DNNs are tailored for data of a fixed size without time dependencies such as images, not sequential data.

3.1.2 Recurrent neural networks

Recurrent neural networks are natural extensions of DNNS, designed to handle sequential or temporal data. RNNs are a better option for AMT applications, since consecutive frames will include both present and past feature. The following transformation is done in each layer.

$$h_{l+1}^{t} = f(W_l^f h_l^t + W_l^r h_l^{t-1} + b_l). \tag{3.2}$$

Parameters W_l^f and W_l^r are the weight matrix from the input to the hidden layer and weight matrix for the recurrent connection respectively. These parameters are calculated by the backpropagation through time algorithm (BPTT) and SGD. RNNs learn dependencies that occurred in the previous step in time. Event though these models are capable of capturing complex dependencies and hence solving difficult problems, they are notoriously hard to train and tricky to use. One of the main limitations of RNN is that they are not able to learn dependencies that are separated by several steps in time due to the well known issue of vanishing gradients[17].

3.1.3 Long short-term memory networks

Long short-term memory networks (LSTM) is a sort of RNNs structure that learn long term dependencies by using the memory cell. LSTMs have replaced traditional RNNs for the majority of sequential series problems.

3.1.4 Convolutional neural networks

Convolutional neural networks(ConvNets), are designed to process data that contain some sort of spatial topology, most likely on a grid, for example a colour image composed of three 2D arrays containing pixel intensities in the three colour channels [13, 7]. Many data modalities are in the form of multiple arrays: 1D for signals and sequences, including language; 2D for images or audio spectrograms; and 3D for video or volumetric images. There are four key ideas behind ConvNets that take advantage of the properties of natural signals: local connections, shared weights, pooling and the use of many layers [12]. ConvNets are composed of alternating convolution and pooling layers, followed by one or more fully connected layers.

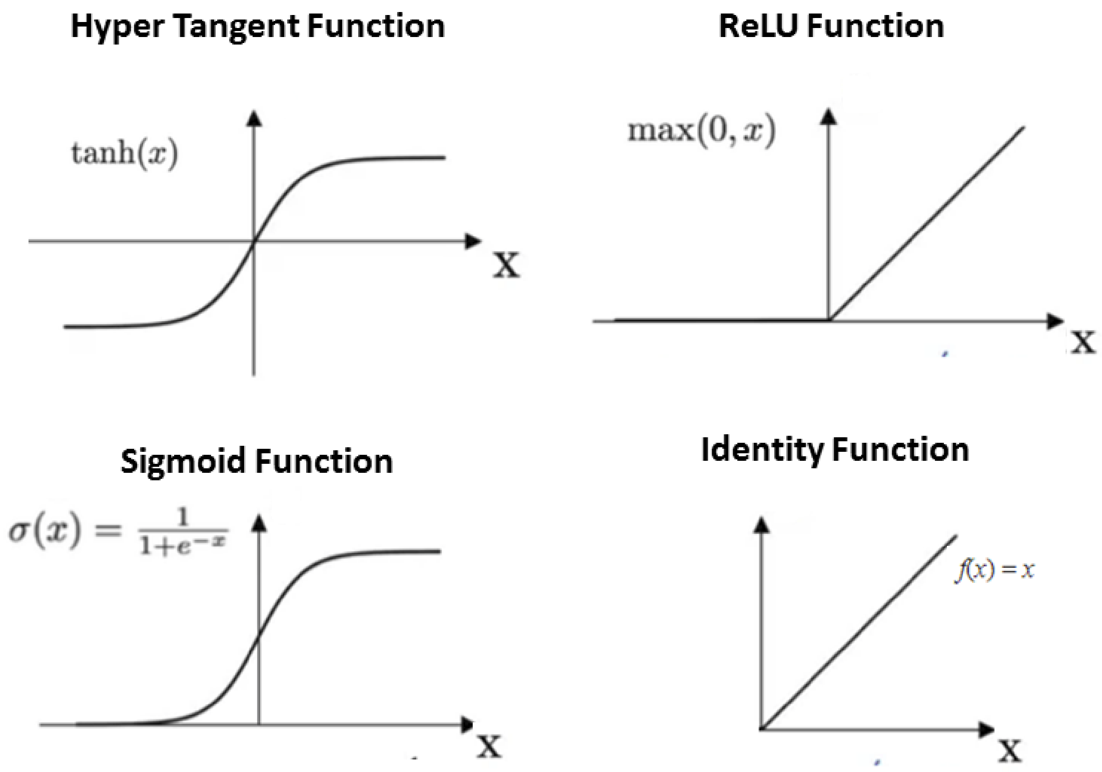


Figure 3.1: Popular Activation Functions

Convolution

Convolutional neural networks get their name because they make extensive use of convolutional layers. Convolutional layers are base on a well known algebraic operation called convolution defined by equation 3.3.

$$s(t) = \int x(a)w(t-a)da \tag{3.3}$$

Formally, the repeated application of the shared weights to the input signal constitutes a convolution operation:

$$h_{j,k} = f(\sum_r W_{r,j}x_{r+k-1} + b_j) \tag{3.4}$$

The parameter h is called feature map. The x is a vector of inputs from different channels, for instance, RGB channels for images and each input x_i has an associated weight matrix. $W_{r,j}$ denotes the weights (parameters) of the convolutions, also known as kernels.

In context of convolutional networks, the input is usually a multidimensional array of data, and the kernel usually have a shape of a multidimensional hypercube (for simplicity, reader can imagine a rectangle or a square). This means that in practice, even though the kernel can be infinite, we set most of its weights to zero. This allows as to efficiently compute the result of the convolution in a finite time.

These convolutional layers turned out to capture the inherent biases of image data extremely well, and right after their proposal improved the state of art in multiple fields.

When convolutions are used in the context of music on a spectrogram, they are often constructed to capture the whole size of the spectrogram for one particular time, so they can be imagined as a long, narrow rectangle applied across the time dimension.

Max pooling

It is a common practice in the design of neural networks to add a max pooling (or pooling in general) layer. This layer accumulates some form of local statistics for it's input and returns it as an output. In our case, the max pooling layer returns just the value of the maximal input in it's field of vision [32], as seen on figure 3.2.

Max pooling is often say to introduce another level of rotation and translation invariance to the neural network.

Dropout

We often add max pooling after every convolutional layer as a form of regularization and to introduce more robustness into the network. In a similar fashion, we make use of a dropout layer to normalize the fully connected layers that are at the end of the network.

This trick was originally proposed for fully connected deep neural networks [24], but found broad usage across multiple different styles of the neural net architectures.

The idea is fairly simple and is illustrated by figure 3.3.

If we imagine the neural network as a set of nodes and connections between them, we simply remove a set of nodes prom particular layer and ignore their value. We decide which nodes to remove randomly during each training step independently for each node by drawing a number from a binomial distribution. The probability used in this distribution is commonly called a dropout rate. Dropout denotes a fraction of the nodes that remains present in the network.

This layer is commonly implemented by multiplying the output vector of a given layer element-wise by a binary vector.

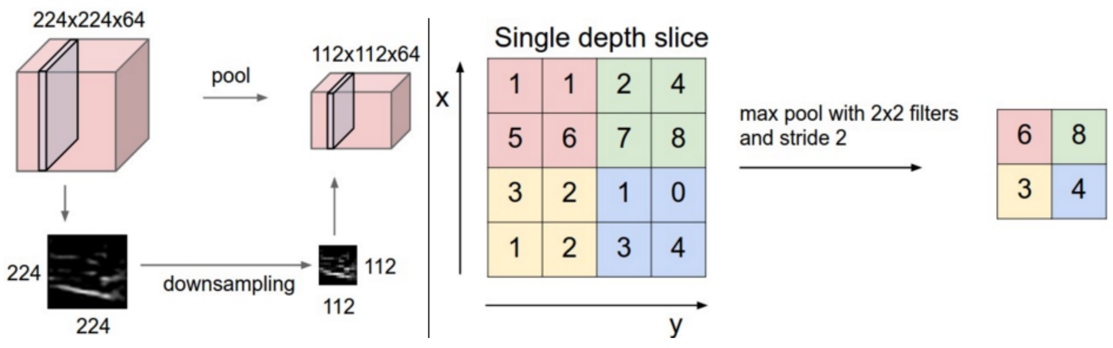


Figure 3.2: Max Pooling 2x2

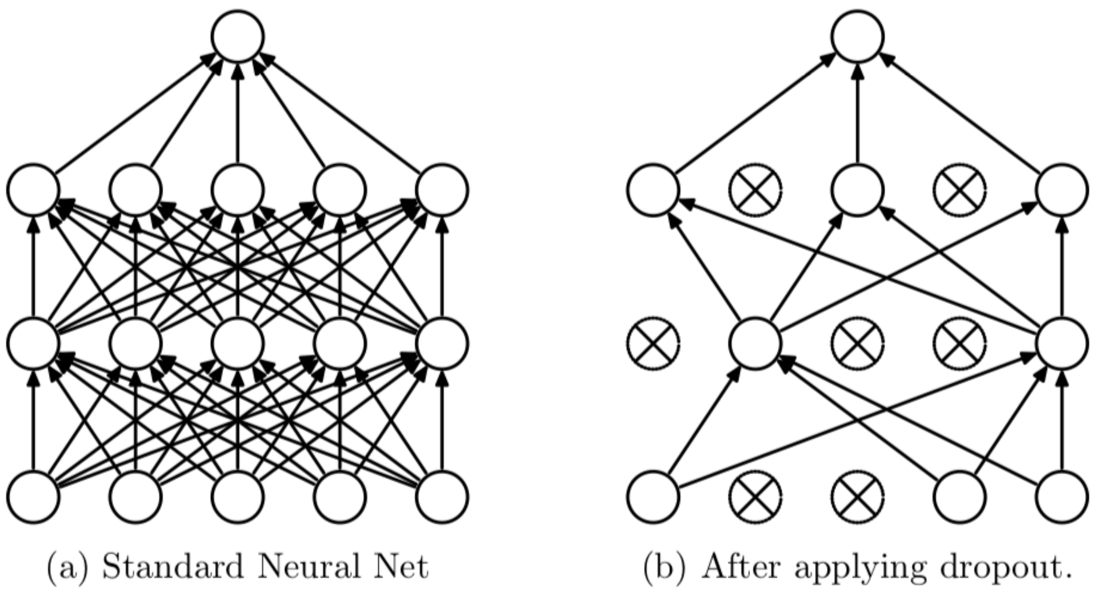


Figure 3.3: Dropout

Batch normalization

Another common technique used to improve the training of deep neural network is batch normalization [10]. Authors refer to a phenomenon of internal covariate shift. However, lately there is a huge discussion in the machine learning community on why does this method work, and what it actually does [21, 31, 14]. However, it turned out to work very well in practice and authors (and other researchers) broke multiple state of the art benchmarks.

Because of this, we only include the equation with basic factual explanation.

$$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \tag{3.5}$$

$$\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \tag{3.6}$$

$$\hat{x_i} \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{3.7}$$

$$y_i \leftarrow \gamma\hat{x_i} + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \tag{3.8}$$

Batch normalization is applied on batch outputs of a ith note in a particular layer x_i. We keep track of statistical properties of the batches that run through the network. We accumulate the mean of the values in μ_B, variance σ_B^2. Then we mean center the values $\hat{x_i}$. Finally we obtain the output of the batch normalization layer y_i by rescalling the $\hat{x_i}$ with γ and β.

Chapter 4

System architecture

In this chapter, we introduce a related work, some possible architectures for music transcription systems based on deep learning and improvements of those architectures. Next, we present preprocessing of our data in a more detail way and some challenges during the training process.

4.1 Related work

We consider a paper An End-to-End Neural Network for Polyphonic Piano Music Transcription published by Siddhart Sigtia, Emmanouil Benetos, and Simon Dixon in 2016 [23], to be our main inspiration.

They were the first to apply supervised neural network models for polyphonic piano automatic music transcription. They used a MAPS dataset and experiment with the constant Q transform (CQT) as the input representation. In the paper, authors explored the use of multiple different acoustic models such as DNN, RNN and ConvNet. They performed an exhaustive search in the space of hyperparameters per each kind of model, such as the number of layers, the number of hidden units, activation functions, optimizers, dropout rates, learning rates, batching and so on. In this work, we take a closer look at the model configurations for the best performing architectures and try to replicate them, and maybe improve upon them. In addition to this, we evaluate some of them on four datasets - MAPS, CPMT, MIMAS and Hybrid dataset.

Since the publication of the work by Sigtia, many other similar papers that were based on his work have been published. One of the most interesting papers is the one published by Google Brain Team in 2018 [8]. This paper describes a deep convolutional and recurrent neural network which is trained to jointly predict onsets and frames. They evaluate input data in the form of mel-scaled spectrograms with log amplitude in contrast to Sigtia work that used a constant Q transform as input representation. They built their own model, which is divided into two parts. The first one is focused on an onset prediction and the second one on the frame prediction. They achieved great results 4.1.

	Frame			Note		
	P	R	$F1$	P	R	$F1$
Sigtia et al., 2016 [23]	71.99	**73.32**	72.22	44.97	49.55	46.58
Kelz et al./ 2016	81.18	65.07	71.60	44.27	61.29	50.94
Melodyne (decay mode)	71.85	50.39	58.57	62.08	48.53	54.02
Onsets and Frames	**88.53**	70.89	**78.30**	**84.24**	**80.67**	**82.29**

Table 4.1: Precision, Recall, and F1 Results for acoustic models trained on synthesised pianos and tested on real recordings.

And the last thing we would like to mention in this section is a commercial product *Anthem score* which is the leading software for automatic music transcription [15]. The transcription system of this software is also based on machine learning models. They use a ResNet architecture, which won the ILSVRC challenge in 2015 and also profoundly focuses on the onset tone prediction. However, architecture hyperparameters are not known. We know they created a dataset of 2.5 million training examples from 3000 MIDI files spanning several different genres of music. So we can consider it to be our inspiration and in this work, we will try to create some hybrid dataset too.

4.2 Proposed Models

4.2.1 DNN Acoustic Model

The first and simplest acoustic model is DNN. It is considered to be the simplest to train and often requires only a small set of hyperparameters. Sigtia performed a grid search over many variations of these hyperparameters and concluded that one particular set of their values as the best one. We use the same set of hyperparameters with small adjustments.

The model has three layers with 125 hidden unit, activations functions such as ReLU and sigmoid, where ReLU is used for hidden layers and sigmoid for the output layer. We use a dropout rate of 0.3 for all layers. We use an Adam optimizer and the learning rate of $1e-2$ that decreasing linearly to 0 over 1000 epochs in contrast to Sigtia, who uses ADADELTA optimizer which is an adaptive learning rate method because it gives us better results. We save the model after each epoch and observe loss over the validation set, and if the loss does not decrease after 20 epochs, we stop the training process. Just like Sigtia, we use mini batches of size 100 for the SGD updates.

Model	DNN	LSTM

Layer 1	Dense(125)
	Dropout(0.3)
	ReLU
		LSTM(200)
		Dropout(0.3)
		tanH

Layer 2	Dense(125)
	Dropout(0.3)
	ReLU
		LSTM(200)
		Dropout(0.3)
		tanH

Layer 3	Dense(125)
	Dropout(0.3)
	ReLU	

Output layer	Dense(88) Sigmoid	Dense(88) Sigmoid
#parameters (36 bins)	74,213	700,888
#parameters (48 bin)	86,713	780,888

Table 4.2: DNN and LSTM architectures

4.2.2 RNN Acoustic Model

As well as in the previous model, we take the best performing configuration from Sigtia and evaluate it. The model has two numbers of hidden layers, with 200 units per layer. The training sequences are further divided into sub-sequences of length 100, and SGD updates are made of one sub-sequence at the time, without any mini batching [23]. Similarly to the previous model, we observe the loss over the validation set, and if it does not decrease after 20 epochs, we stop the training process and set up the learning rate of $1e-2$ that decreases linearly to 0 over 1000 epochs.

4.2.3 ConvNet Acoustic Model

We implement the best performing configuration of the CNN architecture made up by Sigtia. It consists of a few more parameters than previously introduced architectures. As input representation, we do not have one - dimensional vectors like in case DNN but overlapping windows. Our parameter for window size is 7. So we take as an input our normalized spectrogram and create overlapping windows over the spectrogram with this window size. Next, the CNN model contains two convolutional layers with two different window shapes for the convolutional layer. The first layer has the window shape of size (5, 25) and other has the window shape of size (3, 5). The pooling size and dropout rate are fixed for all convolutional layers they are (1, 3) and 0.5, respectively. Also, all convolutional layers correspond to 50 filters per layer. Convolutional layers in the architecture follow two fully connected layers with 1000 hidden units and 200 hidden units, respectively. For the training, a batch size of 256 is used, and early stopping is used to stop training if the validation loss did not decrease after 20 epochs. An initial learning rate of 0.01 is set up and linearly decreased to 0 over 1000 iterations, and a constant momentum rate of 0.9 is used for all the updates.

Model	ConvNet	ConvNet BNR
Layer 1	Conv2D(50x3x35)	Conv2D(50x3x35)
		BatchNormalization
	Dropout(0.5)	Dropout(0.5)
	Tanh	Relu
	MaxPooling(1x3)	MaxPooling(1x3)
Layer 2	Conv2D(50x3x35)	Conv2D(50x3x35)
		BatchNormalization
	Dropout(0.5)	Dropout(0.5)
	Tanh	Relu
	MaxPooling(1x3)	MaxPooling(1x3)
Layer 3	Dense(1000)	Dense(1000)
		BatchNormalization
	Sigmoid	ReLU
	Dropout(0.5)	Dropout(0.5)
Layer 4	Dense(200)	Dense(200)
		BatchNormalization
	Sigmoid	ReLU
	Dropout(0.5)	Dropout(0.5)
Layer 5	Dense(88)	Dense(88)
	Sigmoid	Sigmoid
# Param (36 bins)	1,462,738	1,467,938
# Param (48 bins)	2,012,738	2,017,938

Table 4.3: ConvNet and ConvNet BNR architectures

4.2.4 ConvNet BNR

We noticed that Sigtia does not use ReLU as an activation function in his models. Nowadays, it highly recommended using ReLU for training deep neural networks over sigmoid and tanh functions. Therefore we change sigmoid to ReLU in Sigtia proposed mode ConvNet model in order to get better results. Also, Batch normalization is a commonly used technique for improving speed, performance, and stability of neural networks in nowadays. We decided to combine this technique with ReLU in the hope of outperforming Sigtia original model. Other hyperparameters remain unchanged. In the table 4.3 below, we see a comparison of configurations these models. We called this model ConvNet BNR(Batch Normalization and Relu).

4.3 Preprocessing

Before the start of the training of we must extract features from the audio data in order to make it more understandable by the neural networks.

4.3.1 Audio files

We create a preprocessing script that takes in audio files, as well as matching MIDI files for building input/output pairs for the neural network. The audio files are stored in uncompressed audio format WAV, in lossless compression format FLAC and also in loosely compressed MP3. The content of all these files is a 1-dimensional array of numbers that represents sound pressure over time. For arbitrarily handling of any type of audio files, such as downsampling to a sampling rate of 16000kHz, we use the Librosa library [16].

After the audio samples are loaded into memory, we need to compute a time-frequency representation of the audio. We evaluate both Mel-spectrum representation and constant Q transform (CQT) because it has logarithmically spaced bins which allow us to make training of the neural network much easier since harmonic patterns stay constant for each note, or are pitch invariant. To compute both representations, we use the Librorsa library as well.

For constant Q transformation over 7 octaves, we choose two sets of configuration. In the first one, we use 36 bins per octave as Sigtia used, and in the second, we use 48 bins per octave. A hop size of 512 samples is used for both sets of configuration. In total, we have 252-dimensional and 352-dimensional input vectors of real values. We have 31 frames per second which should be enough to capture even fast changes in the pitch. To normalize our input vectors, we compute the mean and standard deviation of each dimension over the input set and transform the data by subtracting the mean deviation and diving by the standard deviation. This normalized data is then passed as an input to our proposed acoustic models.

For Mel-scaled spectrograms, we use 229 logarithmically-spaced frequency bins, a hop length of 512, to obtain the final 229-dimensional input vector of real values.

4.3.2 Midi files

The MIDI files are our ground truth transcriptions (labels) during the training process. For all necessary work with MIDI files, we use the *pretty_midi* library [18]. This library is characterised by its function of making and reading MIDI files. Here we can gain all the information that is needed, such as note-on(onset) and note-off(offset) events including pitch and tempo, as well as convert our MIDI file into a piano roll or time-note representation.

First, we sample MIDI transcriptions at the same rate as the audio. This is done by a function parameter 'times', that permit sampling MIDI files at specific times. We get these times by knowing the shape of the audio spectrogram, used sampling rate and hop size. We use the Librosa library to compute this. After knowing the 'times' we run $piano_roll$ function with well-know parameter 'times', and then we obtain sequences of 88-dimensional binary vector for our training purpose. These 88 dimensions match the notes A0-C8 on a piano. Now our audio input data perfectly matching with MIDI transcriptions frame by frame.

4.3.3 Technical details

In the end, we save all these inputs vectors into NumPy data files (extension .npy). We choose npy over the txt and csv formats because it is much faster and more efficient way of loading, storing and manipulating these kinds of data.

In total, we deal with a 15GB preprocessed data of CMPT dataset that takes three hours of computing time, 6.1GB preprocessed data of MAPS dataset that takes one hour of computing time and 0.5GB of MIMAS dataset which takes few minutes to compute.

4.4 Training

According to the table 4.4 the length of training corresponds with parameters of models. The DNN acoustic model has something about 80k parameters while ConvNet BNR has 2 million parameters and we see that the training of DNN cost us a much less time as ConvNet training. That means the time depends on the complexity, or amount of parameters of the model.

Also, we tried Mel spectrum spectrograms to train but results were too bad so we decided to not continue with this training process.

Acoustic Model	Total time trained	# epochs	avg epoch time
DNN	1.64hrs	106	56s
ConvNet	5.4hrs	10	31min
ConvNet BNR	5.58hrs	10	33min

Table 4.4: Time for training each acoustic model

4.4.1 Hardware specification

To train and evaluate the proposed models we use Google Colab notebook which has a 12.6 GB available RAM, but after session crash, it can be double up to 25GB RAM memory, 2 core CPU with 2.2GHz, 100GB available disk space. For all computing power, it uses Tesla T4 graphic card. We set up a pipeline of preprocessing, training, post-processing, and evaluation in this colab.

We considered involving Google Cloud Engine to speed up the training process. It is a premium service, but as a new user, you get free 300$ in credit, and it can be easily connected with Google Colaboratory notebook. We wanted to spend all of the credit for 8CPUs, 52GB RAM and NVIDIA Tesla v100 graphic card. Estimated price for this configuration was around 2.6$ per hour. However, creating a process of virtual instances did not work for us because resources with our type of requirements were not available.

4.4.2 Loss function

We use the binary cross-entropy as a loss function since we deal with a multi-label problem.

This is described in the equation 4.1.

$$L = \frac{1}{N} \sum_{n=0}^{N} y_n \ log \ \hat{y_n} + (1 - \hat{y_n}) \ log(1 - y_n) \tag{4.1}$$

4.4.3 Optimizers

During the training process, we noticed that optimizers and learning rates used by Sigtia do not work for us. We replaced the optimizer used by Sigtia with Adam since Adam optimizer has the best performance for all acoustic models in our work. Further, we adjusted the learning rate.

Since we were not able to get close to Sigtia results for DNN architecture, we increased our performance involving Adam optimizer and used a higher input resolution of spectrograms. However, our performance was still not comparable to Sigtia for this acoustic model.

We noticed that using a learning rate of 0.01 and SGD optimizer with a constant momentum rate of 0.9 does not work for us as well as authors claimed. However using Adam optimizer and the learning rate of 0.001, we are able to get closer to the Sigtia results.

4.4.4 Challenges

Working with such a huge amount of data and parameters can lead to many errors. In this section, we would like to point out some of them. For instance, we were not able to fit our whole dataset into the memory, so we needed to read it from the disk in an online fashion, we had problems with colab session expiration and even storing our large amounts of data turned out to not be straight forward. So be aware, if you will deal with work like that the debugging process can be very time-consuming and be sure the data alignment is correct. Here we describe some of our solutions.

Data generator

Since we have to deal with large amounts of data because the sound files are large, and preprocessing and slicing to windows makes them even larger, we need to figure out how to store this data into memory efficiently. We are not able to store this amount of data in the RAM memory at the same time, because of the limited number of resources provided by Colab.

We figured out that Keras provides the Fit Generator, which brings an opportunity to deal with data more efficiently. So now we do not have to load all the data at the same time, but instead, we can do it step by step by freeing up the memory that is no longer being used.

We implemented a DataGeneration class, which allows us to manipulate with only currently used data and efficiently use our resources.

Session expiration

Another problem we encountered was tied to the limitations of the free version of Colab. Session that facilitated our access to the compute power always expired after 90 minutes of user inactivity (even though, the code was still running). This caused our experiments to be killed in the middle of the night.

To get around this, we implemented microbot. It is a short JavaScript program that generates a user activity by clicking inside the Colab in predefined intervals of time, which prevents the session from expiring. This turned out to be a very reliable solution, and we were able to run long experiments without any human supervision.

Storage service

We needed to store it somewhere and make the data accessible for Google Colab. Here we came up with a plan to store all of the data in Google Drive. One of the most significant advantages of Google Drive in comparison to other cloud services is that it directly connected with Google Colab.

Chapter 5

Evaluation

In this chapter, we present our experimental results, post-processing method, and datasets that we used.

5.1 Datasets

To evaluate the perfomance of different approaches, we use of multiple datasets and in this section, we describe these datasets in more detail.

5.1.1 MAPS dataset

As a primary dataset we use for evaluation of our acoustics models is MAPS (MIDI Aligned Piano Sounds) dataset which contains audio and corresponding annotations of isolated notes, chords, and complete piano pieces. This dataset was proposed by Emiya [3]. For us, the most interesting part of the dataset is a set of piano compositions. These consist of 270 pieces of classical piano music with MIDI annotations and are split into 9 categories of different piano types and recording conditions, with 30 recordings per category. There is another split with a ratio 7:2 which says that 7 categories(210 pieces of piano music) are rendered by software synthesizers while remaining 2 sets of audio (60 pieces of piano music in total) are produced by Yamaha Disklavier piano, so it means they are actual recordings of piano performance.

5.1.2 CPMF dataset

`www.piano-midi.de/` is a German that contains scores in MIDI files and corresponding audio files for multiple classical piano pieces. For audio files this website uses lossy compression format mp3.

We construct a Classical piano MIDI files (CPMF) dataset by crawling this website. We wrote a Python script that finds all the piano pieces present on the website, download the MIDI file and corresponding mp3s. This script runs in a matter of few minutes, which allows anyone else to easily recreate this dataset. Note, that the mp3s were originally created by the website authors by running various kinds of synthesizers on the original MIDI files.

In total, this dataset contains 30GB of uncompressed audio files with 22 hours of classical piano music split across 238 distinct performances.

5.1.3 MIMAS dataset

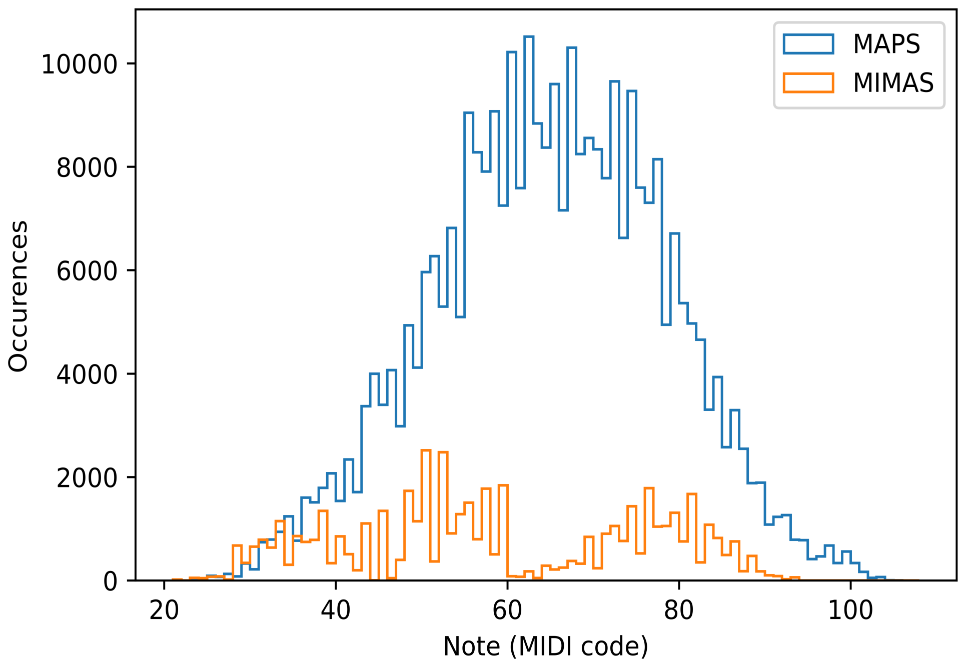
In this work we propose a novel Multi instrument midi aligned sounds dataset focused on traditional Slovak music. It was created in a collaboration with Pavol Škoda and it contains only Slovak folk songs obtained by playing an electronic accordion Roland FR-8x and arranger workstation Korg Pa3X. In audio pieces from this dataset, we can find musical instruments such as accordion, cimbal, viola, contrabass and piano. The dataset is split into two parts by musical instruments included. The first part contains only a piano performance and the other one is a mix of remaining instruments. The main reason for this split is that the piano part dominates and is mainly developed for the training process. On the other hand the mixed part is not very big and serves us mainly for testing and valuation.

For the piano part of this dataset we use a GarageBand application developed by Apple, which is a free music creation studio for all mac users. We connect electronic accordion Roland FR-8x to a computer via MIDI, open Garage Band choose MIDI input and then we can play. It records 3 channels(bass, chords and right hand) to midi and later we use fluidsynth to create the piano sounds.

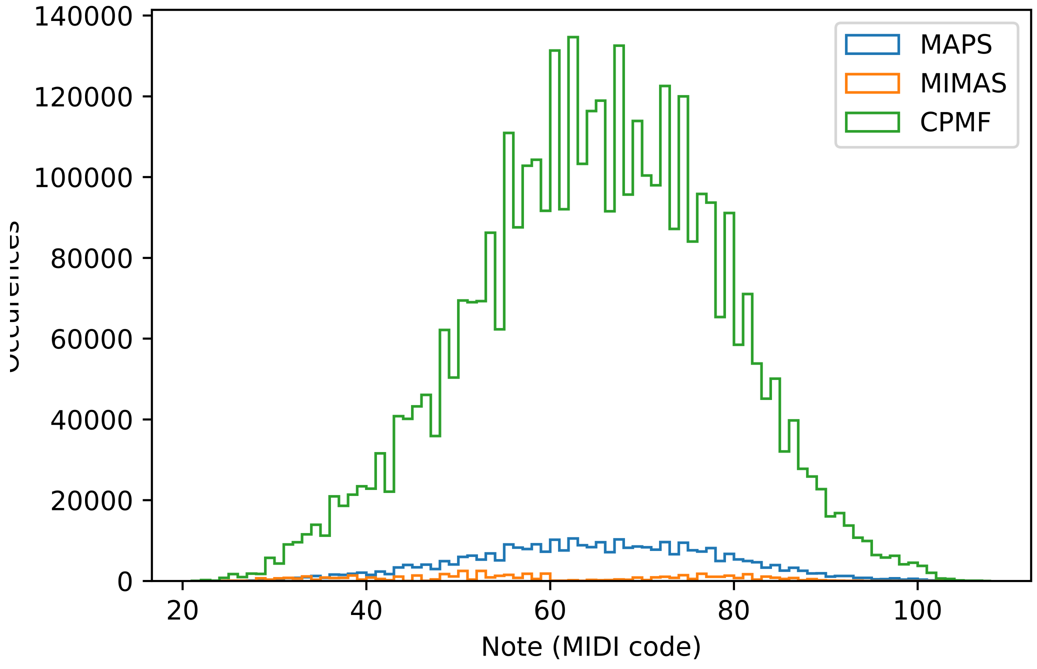
The multi musical instrument part had many more complications. We connected Roland FR-8x to Korg Pa3X via MIDI and recorded 4 channels, but bass, chords and right hand were played on 3 channels at once. The remaining fourth channel was used to change solo instrument in the right hand. At first, we recorded midi by fluidsynth, and mp3 in Adobe audio, but since the audio was not automatically aligned with the midi records we decided to use Cubase which allows it. Behringer XR18 was used as a sound card and MIDI converter.

We recorded both infamous Slovak folklore songs and typically folklore songs per some regions such as Podpoľanie, Novohrad, Gemer, Hont, Horehronie and Šariš. Recordings have a leading solo instrument accompanied by bass and chords. Chords sounds are performed by viola, bass sounds are recorded by double-bass and both cimbal and accordion are used as leading instruments. However, in case of the piano part bass, chords and solo instrument are performed by the piano synthesizer.

We recorded not just a melody of the songs but also different combinations of chords and basses and what is unique is our style of variations played in high temps. In other words, half of the dataset has 4 or even 6 tones per beat that in high tempos is quite a lot and it could be kind of challenging for transcriptions systems.



(a) MAPS and MIMAS Histogram



(b) MAPS, MIMAS and CPMF Notes Histogram

Figure 5.1: Comparison of three of our dataset MAPS, MIMAS and CPMF.

Additionally, we recorded a few separated performance of solo instruments, viola chords and double-bass tones to include this part to the training process in order to slightly increase the performance of our acoustic models.

5.1.4 Hybrid dataset

This dataset is a combination of MAPS, Piano-MIDI and MIMAS. We take 90 pieces of the piano music of MAPS dataset including real piano recordings, 60 pieces of audio of CPMF dataset and 20% of MIMAS including mixed part of it. Purpose of this dataset is to train a hybrid model that can be able to make satisfactory prediction of both classical music and folklore music.

5.2 Testing

5.2.1 Metrics

To measure the performance of our system, we use frame-based metric, and it is made by comparing the transcribed binary output and the MIDI ground truth frame-by-frame. The evaluation for the metric is achieved by calculation F-measure, precision, recall, and accuracy. These parameters are given by the equations 5.1, 5.2, 5.3 and 5.4 as follows

$$Precision(P) = \sum_{t=1}^{N} \frac{TP(t)}{TP(t) + FP(t)} \tag{5.1}$$

$$Recall(R) = \sum_{t=1}^{N} \frac{TP(t}{TP(t) + FN(t)} \tag{5.2}$$

$$Accuracy(A) = \sum_{t=1}^{N} \frac{TP(t}{TP(t) + FP(t) + FN(t)} \tag{5.3}$$

$$F - measure(F) = \frac{2PR}{P + R} \tag{5.4}$$

where TP, FP, and FN stand for the number of true positives, the number of false positives, and the number of false-negative classifications.

5.2.2 Post-proccesing

After the training is done, we perform some cleaning of the predictions. We use thresholds as a post-processing method. The same threshold is picked for each pitch class to determine whether or not it is active. We did not find this threshold by calculating the precision-recall curve for potential thresholds across all notes and choosing one that maximizes both like Sigtia. Instead, we rounded our predictions, and in order to improve accuracy, even more, we filled empty gaps between in concise window. In other words, if we found the active pitch and this same note is active in three, four frames in front of us, then we take a look between these frames, and if we found that this same pitch is not active in this frames, we activate it. The time complexity of this algorithm is $O(n * d * k)$ where n is an input size, d is the length of the window, and k is a note range we predict. Since we deal with an enormous amount of data, the algorithm can slow down during data evaluation, but it is fast enough in the case of individual transcription.

5.2.3 MAPS Evaluation

We successfully trained and evaluated 3 of 4 acoustic models for both CQT configurations (36 bins per octave, and 48 bins per octave).

Evaluation on CQT spectrograms using 36 bins per octave

As the first, we evaluated the DNN model on real recordings of MAPS dataset. The results are not so great as we expected. We used different optimizers and learning rates like were used in Sigtia work because they did not work for us. We did not any additional improvements of DNN model in order to get better results because we do not have many options just change the number of layers or units per layer.

Next, we tested ConvNets models there we used different optimizer and learning rate again as has been used by Sigtia. ConvNets performance was potentially good but not such great like in Sigita work again.

We did not close to Sigtia official results and it cannot be achieved because some essential details of Sigtia work remain unknown. Also, we did only 10 epochs of training for ConvNet since we have not so powerful computing resources. Next, we did not use the same set of test data to evaluate like Sigtia because we did not find one part of this data in MAPS dataset which Sigtia used. So we did not really expect the same results as the Sigtia results but we hoped we got a little bit closer to them. Additionally, in contrast to Sigtia we use different post-processing methods. So it is too many factors why we can not be able to obtain the same result as Sigtia work.

However according to table 5.1 , the best acoustic model in our dataset configuration is our ConvNet BNR whose results are in over roughly 5% above Sigtia ConvNet results of our configuration.

Using a batch normalization and only ReLU activation function had a significant benefit for us in the form of better performance. Still, it is not so a great performance as achieved Sigtia.

Acoustic	Predicted				After post-processing			
Model	*P*	*R*	*F*1	*A*	*P*	*R*	*F*1	*A*
DNN	71.06	41.54	52.43	35.53	73.70	40.68	52.42	35.52
ConvNet	54.06	**75.52**	63.01	46.00	56.16	75.87	64.54	47.65
ConvNet BNR	**81.65**	58.12	**67.90**	**51.40**	**82.78**	57.43	**67.81**	51.30
Sigtia	NaN	NaN	NaN	NaN	72.45	**76.56**	NaN	**58.87**

Table 5.1: Precision, recall, f-measure, and accuracy for acoustic models trained on synthesised pianos and tested on real recordings using 36 bins per octave like Sigtia.

We noticed that post-processing was beneficial only for Sigita ConvNet model which accuracy was increased by 1%.

Evaluation on CQT spectrograms using 48 bins per octave

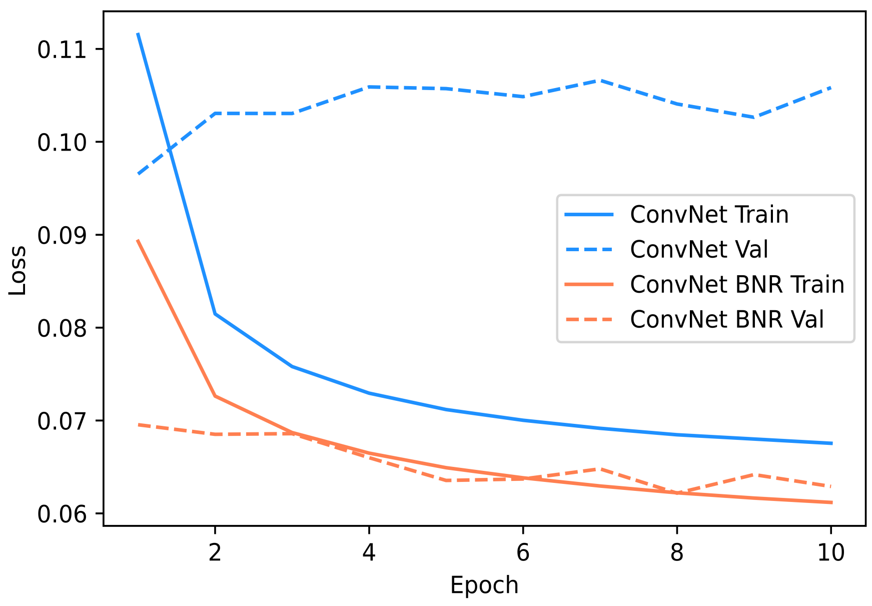
We believed that applied 48 bins per octave in CQT spectrogram can give us better results. We were right. Using 48 bins per octave had significant improvement for each model.

See table 5.2 for results for acoustic models.

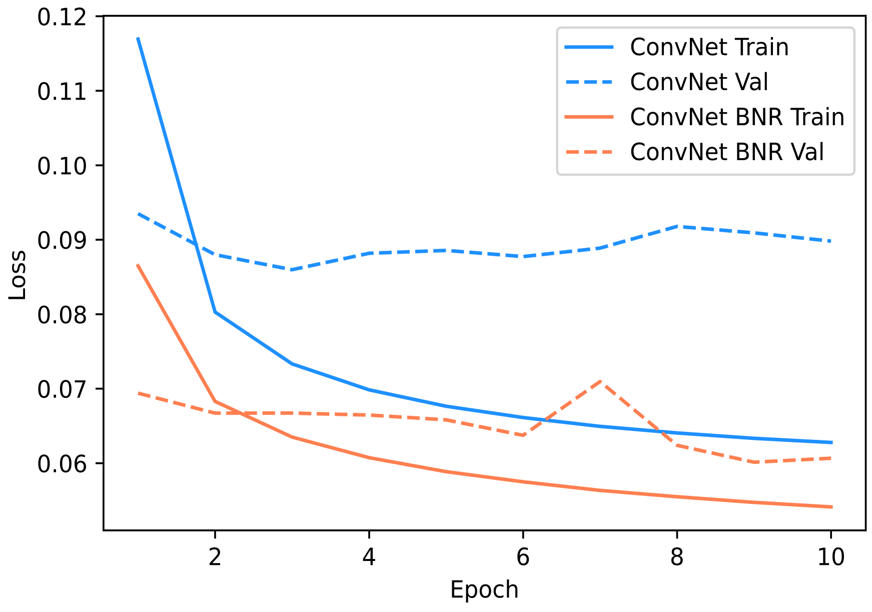
Acoustic	Predicted				After post-processing			
Model	*P*	*R*	*F*1	*A*	*P*	*R*	*F*1	*A*
DNN	75.06	43.36	55.16	38.09	77.82	42.92	55.33	38.24
ConvNet	60.21	**76.28**	67.30	50.71	62.11	**76.56**	68.58	52.18
ConvNet BNR	**81.82**	63.85	**71.73**	**55.92**	**82.27**	63.59	**71.92**	**56.15**

Table 5.2: Precision, recall, f-measure, and accuracy for acoustic models trained on synthesised pianos and tested on real recordings using 48 bins per octave.

The picture 5.2 shows that newly recommended techniques for deep learning are useful and genuinely work very well.



(a) Training and validation loss during the first 10 epoch of training using 36 bins per octave.



(b) Training and validation loss during the first 10 epoch of training using 48 bins per octave.

Figure 5.2: Comparison of ConvNet and ConvNet BNR by using different number of bins per octave

See table5.3 for a comparison of our reproduced models and Sigtia official model. Our reproduced Sigtia ConvNet is worse by more than 10% compared to the original

	DNN	ConvNet	ConvNet BNR	Sigtia
CQT - 36 bins	35.52	47.65	51.30	**58.87**
CQT - 48 bins	38.24	52.18	**56.15**	NaN

Table 5.3: Accuracy for acoustic models trained on synthesised pianos and tested on real recordings.

one. But, we see significant improvement using 48 bins per octave in CQT spectrograms and Batch Normalization and ReLU activation function.

Totally we increase DNN acoustic model performance by 3% and in the case of ConvNet and ConvNetBNR it increase by 5%.

We have not able to replicate the LSTM acoustic model preposed by Sigtia. During the training process, we tried different combinations of optimizers and learning rates but we did not see any improvements. The one epoch took a roughly 41 minutes, and after a few epochs of training the predictions of the model were unsatisfactory and we do not introduce them at work. When we plotted the posteriograms of predictions we noticed that we are not able to train this model and we decided to stop with evaluation and training this model. We do not say that the LSTM acoustic model is wrong because we have not been able to reproduce it. We only need more time and resources to do it.

5.2.4 CPMF Evaluation

For NPMF evaluation, we choose only to use ConvNet acoustic models. Without a doubt, ConvNet BNR is above the Sigtia ConvNet again. It is clearly seen in the picture 5.3 below. However, the metric is not as good as in MAPS dataset. We came up with some possibilities. For instance, these piano midi files are not officially marked as the piano dataset for machine learning problems or music information retrieval. The data can be aligned incorrectly, or the dataset contains is so many combinations of multiple tones that while we do not train over the whole dataset we can not expect good results.

Table 5.4 show us metric results and that post-processing did not improve results, but on the contrary, make it worse.

Acoustic	Predicted				After post-processing			
model	P	R	$F1$	A	P	R	$F1$	A
ConvNet BNR	94.51	20.48	33.66	20.24	94.85	18.74	31.30	18.50

Table 5.4: Precision, recall, f-measure, and accuracy for acoustic model trained on synthesised pianos and tested on real recordings using 48 bins per octave.

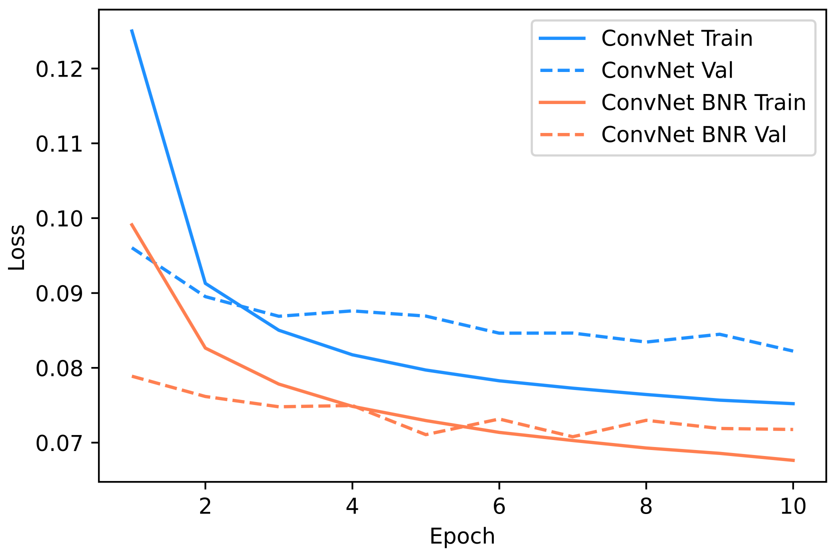


Figure 5.3: Comparison of Sigtia ConvNet and our ConvNet BNR by loss of both training and validation of CPMF dataset using 48bins per octave.

5.2.5 MIMAS Evaluation

In this part of our work which is focused on Slovak folklore songs, we use only a our CovnNet BNR acoustic model because we think it is a waste of time to train the previous models since ConvNet BNR clearly gives us the best transcriptions.

We considered three kinds of possibilities to evaluate this dataset well. At first, we took trained a ConvNet BNR model and tested on mixed part of this dataset. As a second option, we took trained ConvNet BNR model again and did additional training with MIMAS training part. The last option we train a new ConvNet BNR model on our Hybrid dataset.

In table 5.5 bellow we can see results of three our acoustic models tested on mixed part of MIMAS dataset. In this section we do not present post-processing results since they give not any kind of improvement.

Acoustic model	Predicted			
	P	R	$F1$	A
ConvNet BNR (piano)	**51.99**	8.27	14.27	7.68
ConvNet (MAPS)	16.34	**38.85**	23.00	12.99
ConvNet BNR (MAPS)	34.28	29.81	**31.89**	**18.97**

Table 5.5: Precision, recall, f-measure, and accuracy for acoustic models trained on synthesised pianos and tested on MIMAS using 48 bins per octave.

The ConvNet BNR trained on MAPS dataset do the best predictions. We need to notice that Sigtia ConvNet is very close to ConvNet BNR predictions. In the case of MAPS dataset, the difference was roughly 5% in accuracy but here we can see that it is just 3%. The last one ConvNet BNR trained on CMPT dataset obtained the worse its accuracy is only roughly 7%. Any of these three models do not predict satisfactory predictions for MIMAS dataset as we expected. This is caused by combinations of tones collected in MIMAS dataset are distinct to combinations of tones of MAPS dataset.

After we did some additional training on MIMAS dataset of trained ConvNet BNR acoustic model, we evaluated it and given results were pretty scary. Prediction on MIMAS did not improve and relatively good performance on MAPS decreased. We explored the space of possibilities to improve it but we did not succeed in this task. Hence we moved to the third option and it was training new ConvNet BNR acoustic model on Hybrid dataset.

During the traing of ConvNet BNR on our Hybrid datset we experimented with different values of learning rates such as 0.01, 0.001 or 0.0006. However, our ConvNet BNR trained on the Hybrid dataset does not perform as well as we expected. The predictions for MIMAS are little worse as in case ConvNet BNR acoustic model trained on MAPS dataset but predictions tested on the real piano of MAPS dataset are much more worse. We can see the results in table 5.6.

	Predicted			
Dataset	P	R	$F1$	A
MAPS	**94.42**	**27.05**	**42.06**	**26.63**
MIMAS	69.72	19.52	30.40	17.93

Table 5.6: Precision, recall, f-measure, and accuracy for ConvNet BNR acoustic models pretrained on Hybrid dataset tested on both MIMAS and MAPS datasets using 48 bins per octave.

We were hoping that this Hybrid dataset will improve and create the best possible performance for ConvNet BNR acoustic model but it did not happen. We considered some possible improvements and we think that expansion of Hybrid dataset could be the right choice. It is very time- consuming and performance-intensive to evaluate such a large dataset. Therefore we decided it would be the best to try it in our future paper, when we have more computing power available and our MIMAS dataset is big enough.

In the end, we have to point out one important piece of information about MAPS dataset we forgot to mention in the previous chapters. The songs in MAPS dataset repeat and they are performed by another synthesizer or real piano. In other words, when we evaluate training on a real piano part of MAPS dataset, the acoustic model already knows this combination of tones but in an another timbre. This is why the

MAPS results are above others.

In our dataset we do not repeat the same songs, and if we do then we modulate song to another scale or change velocity. There are no possibilities to be close to MAPS results since they are repeating songs and we do not.

Conclusion

In this thesis, we presented the acoustic models for transcribing polyphonic piano performance and Slovak folklore music using deep neural networks. We created and introduced a MIMAS dataset which gives us a great opportunity to explore this field of study.

We tried to replicate Sigtia's work but did not reach their claimed performances. Some of Sigtia hyperparameters for acoustic models did not work for us so we changed the configurations a little bit. We modified Sigita ConvNet acoustic model by adding Batch Normalization and using only ReLU as an activation function. In our work, this model had the best performance among all configurations ran by us, but we were not able to outperform Sigita's original ConvNet in all the measured metrics. We improved their model in some metric, for instance our approaches result in over 10% improvement of precision and it can be significant for some kind of applications. Next, we evaluated the model on our new MIMAS dataset and our ConvNet BNR trained on MAPS dataset gave us the best results.

The most relevant information of a musical note or other sound is onset. One potential improvement is the development of the model which is able to predict only the onset of pitches. Then we can join our ConvNet BNR with this onset model in order to get much better performance. Additionally, we can develop the model focusing on offset predictions of the pitch. Also, we would like to extend our MIMAS dataset.

To conclude, we fulfilled all our goals.

Bibliography

[1] Emmanouil Benetos, Simon Dixon, Zhiyao Duan, and Sebastian Ewert. Automatic music transcription: An overview. *IEEE Signal Processing Magazine*, 36(1):20–30, 2018.

[2] Ronald Newbold Bracewell and Ronald N Bracewell. *The Fourier transform and its applications*, volume 31999. McGraw-Hill New York, 1986.

[3] 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(6):1643–1654, 2009.

[4] Valentin Emiya, Nancy Bertin, Bertrand David, and Roland Badeau. Maps-a piano database for multipitch estimation and automatic transcription of music. 2010.

[5] Geeks for Geeks. An introduction to machine learning. https://www.geeksforgeeks.org/introduction-machine-learning. [Online; accessed 29-February-2020].

[6] Rohith Gandhi. Support vector machine—introduction to machine learning algorithms. *TTowards Data Science*, 2018.

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[8] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck. Onsets and frames: Dual-objective piano transcription. *arXiv preprint arXiv:1710.11153*, 2017.

[9] Jay Hodgson. *Understanding records: A field guide to recording practice*. Bloomsbury Publishing, 2010.

[10] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[11] Anssi Klapuri and Manuel Davy. *Signal processing methods for music transcription*. Springer Science & Business Media, 2007.

[12] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[13] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[14] Xiang Li, Shuo Chen, Xiaolin Hu, and Jian Yang. Understanding the disharmony between dropout and batch normalization by variance shift. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[15] Lunaverus. Cnn. https://www.lunaverus.com/cnn. [Online; accessed 29-February-2020].

[16] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, volume 8, 2015.

[17] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318, 2013.

[18] Colin Raffel and Daniel PW Ellis. Intuitive analysis, creation and manipulation of midi data with pretty midi. In *15th International Society for Music Information Retrieval Conference Late Breaking and Demo Papers*, pages 84–93, 2014.

[19] Sumit Saha. A comprehensive guide to convolutional neural networks—the eli5 way. *Towards Data Science*, 15, 2018.

[20] Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

[21] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 2483–2493. Curran Associates, Inc., 2018.

[22] Tarang Shah. About train, validation and test sets in machine learning. *Towards Data Science*, 2017.

[23] 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.

[24] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[25] Andrew Swift. A brief introduction to midi. *URL http://www. doc. ic. ac. uk/˜ nd/surprise_97/journal/vol1/aps2*, 6, 1997.

[26] Quan Wang, Hannah Muckenhirn, Kevin Wilson, Prashant Sridhar, Zelin Wu, John R. Hershey, Rif A. Saurous, Ron J. Weiss, Ye Jia, and Ignacio Lopez Moreno. VoiceFilter: Targeted Voice Separation by Speaker-Conditioned Spectrogram Masking. In *Proc. Interspeech 2019*, pages 2728–2732, 2019.

[27] Wikipedia. Digital audio — Wikipedia, the free encyclopedia, 2020. [Online; accessed 15-January-2020].

[28] Wikipedia. Fast Fourier transform — Wikipedia, the free encyclopedia, 2020. [Online; accessed 29-February-2020].

[29] Wikipedia. Mel-frequency cepstrum — Wikipedia, the free encyclopedia, 2020. [Online; accessed 12-January-2020].

[30] Wikipedia. Musical note — Wikipedia, the free encyclopedia, 2020. [Online; accessed 23-January-2020].

[31] Greg Yang, Jeffrey Pennington, Vinay Rao, Jascha Sohl-Dickstein, and Samuel S Schoenholz. A mean field theory of batch normalization. *arXiv preprint arXiv:1902.08129*, 2019.

[32] Yi-Tong Zhou and Rama Chellappa. Computation of optical flow using a neural network. In *IEEE International Conference on Neural Networks*, volume 1998, pages 71–78, 1988.

Appendix A: source code

Source code to this bachelor thesis with description and link to the datasets is available on GitHub: https://github.com/Bumaza/Automatic-music-transcription