Automated Music Transcription

using Convolutional Neural Networks

Cota Ionas-Calin

A thesis presented for the degree of
Bachelor of Computer Science

Computer Science
Babes Bolyai University
Romania
4/20/2019

Contents

List of Figures

List of Tables

1 Introduction

Music is universal and its significance is nothing to mess about. Every known civilization has a form of music. From baroque, classical, opera, jazz, traditional folk, rock, rap or contemporary pop music, the sharing of music provides endless joy for humans.

Music transcription is defined as the task of converting music from sound into a written, abstract notation. It is the inverse operation of music performance, which often involves a performer reading music notation and producing sound waves with the help of an instrument or their voice. Because music is an universal language people around the globe can share music with each other surpassing the language barrier.

Manual music transcription is a task difficult enough that even the best musicians struggle to achieve 100% accuracy. It takes a lot of time and practice to learn and even more to master. In order to facilitate the process and make transcription available to everyone people tried to figure out ways to automate the process. It was at this moment, in 1977, Automatic music transcription (AMT) was born.

For the past decades, this field of computer science research has been developing and still has numerous unsolved problems. Every year shows new research with improved algorithms for various sub tasks of AMT.

The goal of this thesis is to introduce the field of automatic music transcription and provide a deep learning approach to pitch estimation using convolutional neural networks (CNNs).

Add brief summary of the application

2 Theoretical Background

2.1 Introduction to Deep Learning

Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to effectively perform a specific task without using explicit instructions, relying on patterns and inference instead. Machine learning algorithms build a mathematical model of sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task. Machine learning algorithms are used in a wide variety of applications, such as email filtering, and computer vision, where it is infeasible to develop an algorithm of specific instructions for performing the task. The represantation of data fed into a ML algorithm plays a major role, as it affects the algorithm's ability to efficiently extract signals and make decisions. Thus, it is important to carefully select the information included in such a representation. Formally, the representation is com- posed of multiple features extracted from raw data. The process of creating new features requires good and intuitive understanding of the data at hand, becoming incrementally time-consuming with the sophistication of the new features. Thus, the biggest challenge of handcrafted features is deciding which features are important and relevant to the problem [2]

2.2 Neural Networks

This section introduces the main concepts related to neural networks. Neural networks have been around since the 1940s and could initially handle only one hidden layer. But with the development of technologies and hardware it became possible to build deeper, more effective architectures, which leads to deep learning as we know it today.

2.2.1 Brief History

At first, neural networks were inspired by how the biological brain works, which is why deep learning was also called artificial neural networks (ANNs)[2]. In biology, a neuron is the cell that receives, processes and transmits information to other neurons through connections called synapses [3]. On the other hand, artificial neurons are defined as computational units (usually mathematical functions) that take one or more inputs and generate an output.

McCulloch and Pits designed an initial version of the neuron as a linear model in 1943, aiming to replicate brain function [4]:

$$f(x, w) = x1 * w1 + x2 * w2 + ... + xn * wn \tag{1}$$

where $x_1, ..., x_n$ are the input values and $w_1, ..., w_2$ is a set of hand-chosen weights.

2.2.2 Components of an artificial neural network

A simple artificial neural network (ANN) consists of input layer, hidden layer and output layer, where the values of the hidden layer are used as inputs for the output layer. A network with several layers is known as a deep neural network. Data flows through the neurons of the layer. Each neuron transforms the input it receives and sends it to the next layer. The neurons share the same characteristics regardless of the layer they are part of.

The Neuron, also called node, is the basic unit of a neural network. Its main components include inputs, weights, activation function and output(s). From a high level point of view the inputs are multiplied by weights, then an activation function is applied to the result and finally, another function computes the output[5][6].

- Weights are defined as adaptive coefficients, whose values are changed during the learning process. They represent the strength of the connection between units. A weight decides how much impact the input will have on the output.

- The summation function helps combine the input and weights, before passing the result to the activation function. Denote the input as $X = [x_1, x_2, ...x_n]$ and the weight vector as $W = [w_1, w_2, ...w_n]$.
  The summation function could be defined as the dot product between these two vectors:

  $$X \cdot W = x_1 \cdot w_1 + x_2 \cdot w_2 + ... + x_n \cdot w_n \tag{2}$$

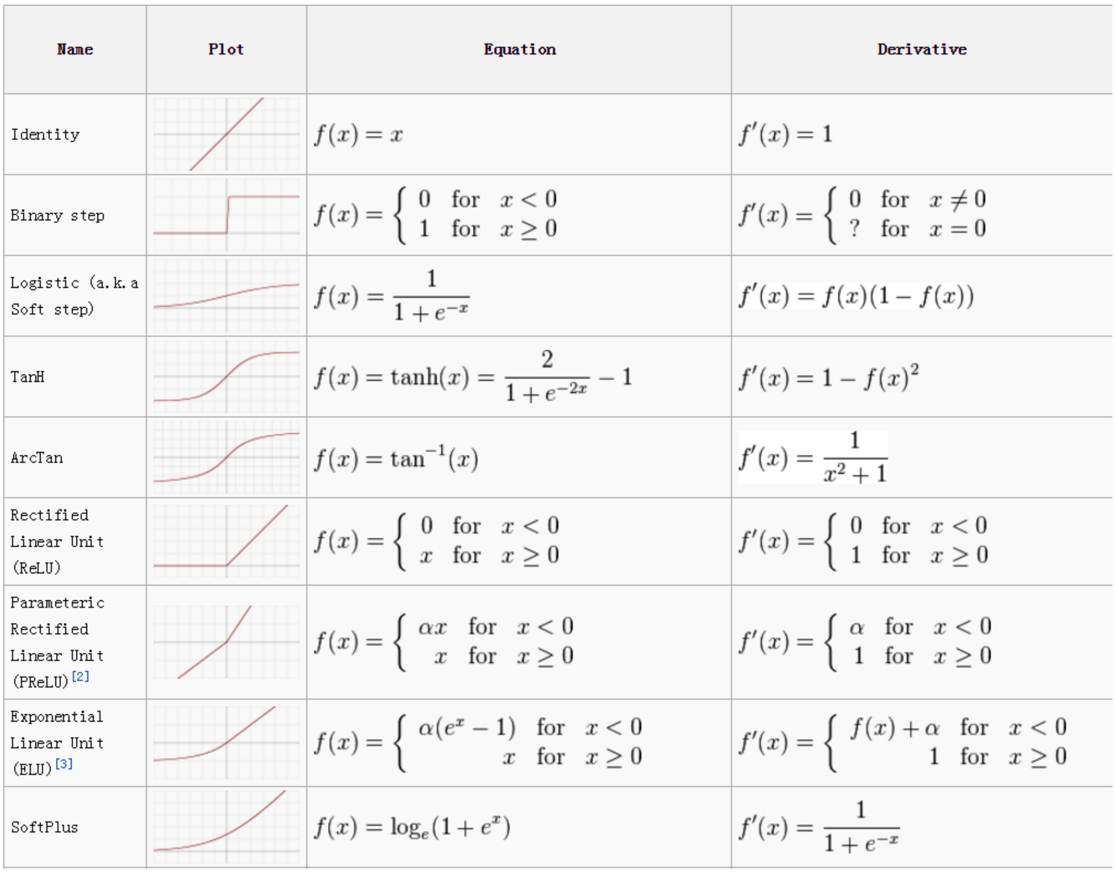
  The summation function could instead compute the minimum, maximum etc. depending on the designated network architecture.The simplest form of an artificial neuron is a linear function which computes the weighted sum of inputs, to which, optionally, bias can be added:

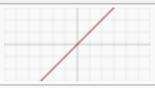
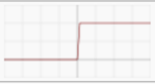
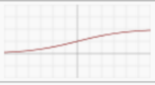
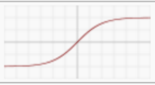
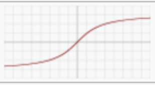
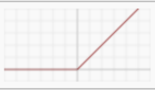
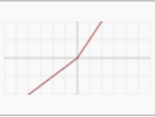
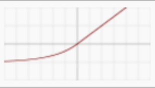
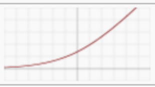
  $$y = \sum_{i=1}^{i=n} (x_i \cdot w_i) + b, \text{ where b is the bias, } x_i \in X, w_i \in W \tag{3}$$

- The activation function transforms the result of the summation function (usually) in a non-linear way. Typically, it has a squashing effect. It serves as a threshold. It divides the original space into two partitions. It's main purpose is to make the neural network non-linear. We denote the activation function as g.

$$y = g(\sum_{i=1}^{i=n}(x_i \cdot w_i) + b), \text{ where b is the bias, } x_i \in X, w_i \in W \tag{4}$$

Figure 1: Common activation functions [7]



Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU)[2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU)[3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

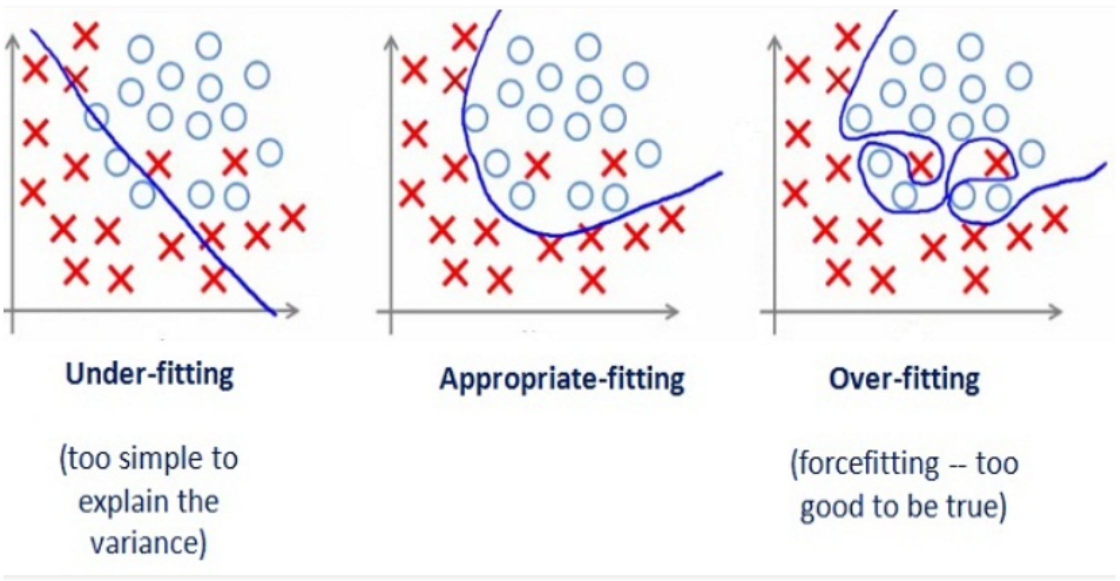
- The output is usually the result of an activation function.

2.2.3 Under-fitting and Over-fitting

Neural networks are able to learn complicated non-linear functions to fit any training set. On the downside, this may lead to over-fitting where the neural network learns the training data so well that it is unable to generalize on new, unseen data. This problem can especially occur on datasets with a small amount of data to learn from.

Under-fitting, the counterpart of over-fitting, happens when a machine learning model isn't complex enough to accurately capture relationships between a dataset's features and a target variable. An under-fitted model results in problematic outcomes on new data, or data that it wasn't trained on, and many times performs poorly even on training data.

Figure 2: Example of over-fitting and under-fitting [8]

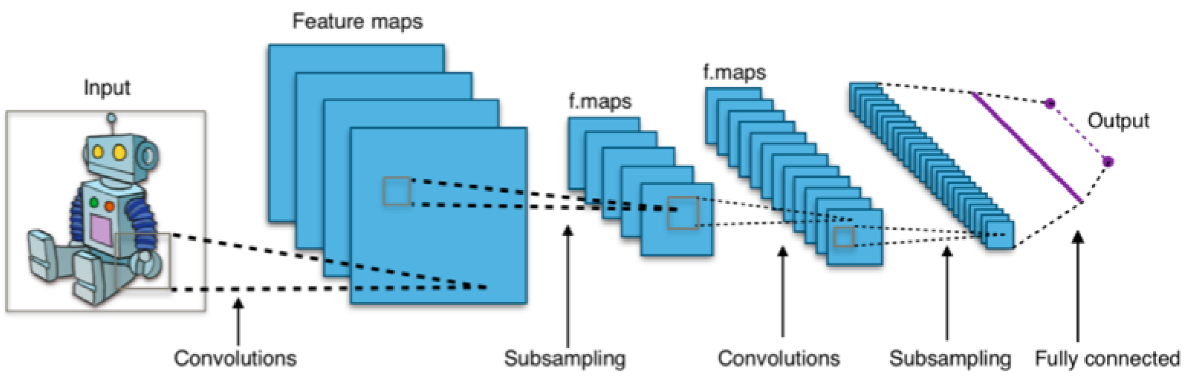

Under-fitting
(too simple to explain the variance)
Appropriate-fitting
Over-fitting
(forcefitting -- too good to be true)

2.2.4 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of Deep Neural Networks, specialized in analyzing images. They were inspired by biological processes. The connectivity pattern between neurons resembles the animal visual cortex. [9]

A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers are typically convolutional layers, RELU layers, pooling layers, fully connected layers and normalization layers. [11]

Figure 3: Typical CNN architecture [10]



- **Convolutional** layers are the core building block of a Convolutional network. It does most of the computation
- **Pooling** layers reduce the dimension of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer
- **Normalization** layers apply a transformation that maintains the mean activation close to 0 and the activation standard deviation close to 1
- **Fully connected** layers link every neuron in one layer to every neuron in another layer
- **Dropout** layers have a % chance to deactivate the neurons of a particular layer. This is a technique used to improve over-fitting by improving generalization. Forcing neurons to deactivate forces the network to learn the same "concept" with different neurons. Mostly used after fully connected or pooling layers

2.3 MIDI format

add some info about the midi file format

2.4 Digital Signal Processing

Digital Signal processing (DSP) is an engineering field focused on analyzing and altering digital signals. It takes real-world signals like voice, audio, video and then mathematically manipulates them. [12]

Signals need to be processed so that the information they contain can be displayed, analyzed or converted to another type of signal. Analog-to-Digital converters take signals from the real-world and turns them into binary digital format. At this point the DSP takes over by capturing the digitized information and processes it, later to be fed back for use in the real-world.

2.4.1 Sound

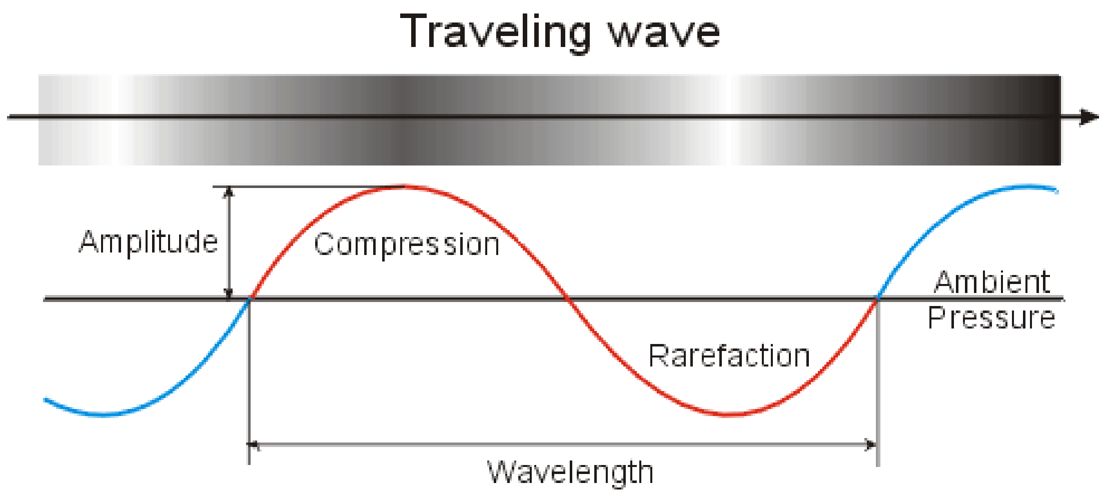
Sound is produced when something vibrates. The vibration causes the medium around it to vibrate as well. Vibrations in the air are called traveling longitudinal waves [13], which we can hear. A sound wave is made out of two areas of high and low pressure called compressions and rarefactions (figure 3).

The pattern of the wave repeats after one wavelength. The height of the wave is called Amplitude. It is what determines how loud the sound will be, the greater the louder.

The wavelength and the speed of the wave determine the pitch (frequency of the sound).

$$c = f \cdot \lambda, \text{ where } c = speed,\ f = frequency,\ \lambda = wavelength \tag{5}$$

Figure 4: Traveling wave components [14]



2.4.2 Pitch

In music, the pitch tells how low or high a note is. In physics, it's measured in a unit called Hertz (Hz) and it's known as frequency. A note that vibrates at 256Hz will be caused by a sound wave vibrating at 256 times/second.

The speed is influenced by the medium in which it travels. Under standard temperature and pressure sound's speed is 343 meters per second. [15]

The equation (5) can be rewritten as:

$$f = \frac{c}{\lambda}, \text{ where } c = speed,\ f = frequency,\ \lambda = wavelength \tag{6}$$

2.4.3 Discrete Fourier Transformation

The Discrete Fourier Transformation (DFT) is one of the most important operation of DSP. It is any quantity or signal that varies over time, such as the pressure of a sound wave, sampled over a finite time interval (often defined by a window function). [16]

$$X[k] = \frac{1}{N}\sum_{j=0}^{N-1}(x[j] \cdot e^{-j\cdot(\frac{2\pi}{N}))\cdot n\cdot k} \text{ for k = 0...N-1} \tag{7}$$

The DFT tells you what frequencies are present in your signal and in what proportions.

It has a complexity of $O(n^2)$ so in practice the Fast Fourier Transform (FFT) algorithm is used instead. FFT runs in $O(n \cdot log(n))$

2.4.4 Fast Fourier Transform

Add more info about fast fourier transform

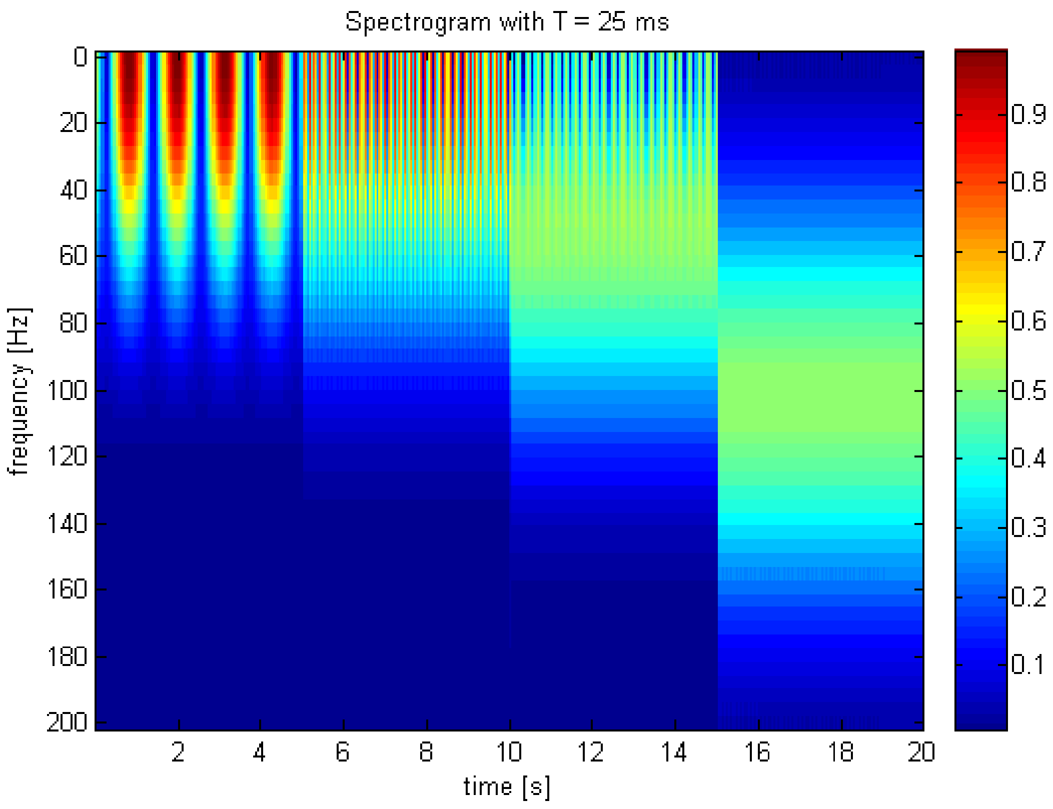
2.4.5 Short-Term Fourier Transform

While DFT is really good by itself, if used on an entire song it would only tell what frequencies exist, but not when they occur. This is where Short-Term Fourier Transform (STFT) comes in handy. It computes DFT over a full signal, but in small segments. Because of this we can see how frequencies change over time, which makes it a good way to compute spectrograms. (Figure 4)

2.4.6 Constant-Q Transform

In general, the transform is well suited to musical data, and this can be seen in some of its advantages compared to the fast Fourier transform. As the output of

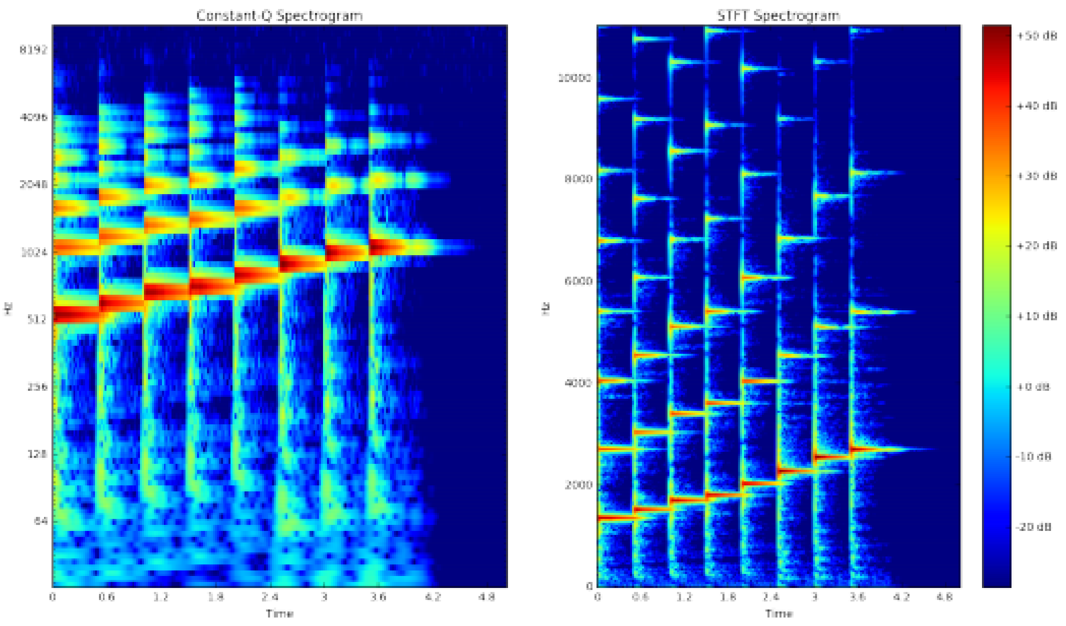
Figure 5: Spectrogram using STFT [17]



the transform is effectively amplitude/phase against log frequency, fewer frequency bins are required to cover a given range effectively, and this proves useful where frequencies span several octaves. As the range of human hearing covers approximately ten octaves from 20 Hz to around 20 kHz, this reduction in output data is significant. [18]

See Figure 5 for a comparison between Constant-Q transform and STFT.

Figure 6: Constant Q (left) vs STFT (right) spectrogram of C major scale



2.5 Music Transcription

In music, transcription is the process of creating a music sheet from a piece or sound. The sheet contains music notation, which consists of different symbols that can be interpreted by musicians, hence it is important for various reasons. Without it composers such as Mozart and Beethoven couldn't have passed their masterpieces across generations. In modern days it helps musicians play songs they never heard before. It's also universal so even if two musicians don't speak the same language, they can read the same notation.

2.5.1 Traditional Music Transcription

In the beginning transcription was done by humans. It's also called musical dictation in ear training pedagogy. [19] It is a skill by which musicians learn to identify pitches, intervals, melody, chords and other elements of music solely by hearing. It is a really hard skill, requires serious training and study and even the best don't have 100% accuracy.

There are some tools to help with the process:

- Musical instruments, help musicians test for certain sounds, trying to mimic what they hear

- Tape recorders
- Nowadays software

Music transcription can be especially difficult and time consuming when the recordings have many overlapping pitches. [20]. The difficulty of this task can be understood in comparison to the ease with which humans can read passages of text and the difficulty of writing down what someone is saying. Another thing that adds to the complexity is that humans often process pitch relatively, rather than absolutely. There are some humans with perfect pitch, which is the ability to recognize pitch in isolation. For those without it, the best approach is guess-and-check method, which is extremely time consuming.

2.5.2 Automatic Music Transcription

The term "Automatic Music Transcription" was used for the first time in 1977, by audio researchers James A. Moorer, Martin Piszczalski, and Bernard Galler [21]. With their knowledge about digital engineering they believed that computers could be programmed to analyze digital recordings of songs such that they could identify things like rhythm, melodies, pitch, bass lines. It's not an easy task. For more than three decades researchers have been trying to crack it open.

Fundamentally, AMT is about identifying the pitch and duration of played notes, so they can be converted in traditional music notation on a sheet.

It has many advantages over traditional transcription:

- Aids experienced musicians in the process of transcribing pieces, increasing their accuracy.
- Makes music transcription available to more people, especially beginners, giving them a chance to share their ideas with others.
- Helps people learn new songs. There are a lot of music sheets online that are not free.
- It speeds up the process. Manual transcription takes a lot of time.

The sub tasks of Automatic Music Transcription:

- **Pitch Estimation**

  There are two versions of this problem. Single pitch and multi-pitch estimation. The real challenge lies in the latter. It's still un unsolved problem [22]. The best algorithms were able to achieve around 70% accuracy, as of 2017. [20]

- **Beat detection**

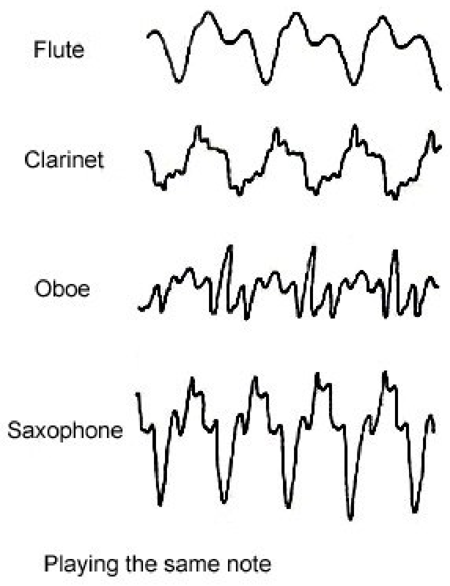
  Beat tracking is the determination of a repeating time interval between perceived pulses in music. [21] Songs are frequently measured for their Beats Per Minute (BPM) in determining the tempo of the music, whether it's fast or slow. Beat can be described as "foot tapping" or "hand clapping" in time with the music. Despite the intuitive nature of the former, which most humans are capable of, developing an algorithm to detect those beats it's difficult.

- **Instrument detection**

  Given an audio recording, the goal is to identify the musical instrument(s) playing each note. Like pitch estimation this problem faces the same monophonic and polyphonic problems, with the latter still unsolved. [23]. This is often simplified to classifying a recording in terms of instrument family, not between specific instruments.

  The most common strategy is in evaluation of various features present in a note as its harmonic shape develops over time. These features are different for instrument families. (Figure 7)

Figure 7: Waveform differences between instruments [24]



2.5.3 Semi-Automatic Music Transcription

In between traditional and automatic music transcription comes semi-automatic music transcription, also called user-assisted transcription. It is a system in which the user provides a certain amount of information about the recording which can be used to guide the transcription process. [25]

For certain use-cases semi-automatic transcription is better than the others as it's faster and more accurate than manual transcription and more practical than the fully automatic. Where semi-automatic transcriptions falls short is databases too large to be done by hand, so they would require way too much user input. With such projects fully automatic transcription is the only way to go.

3 Related Work

Automatic music transcription (AMT) has been attempted since the 1970s and polyphonic music transcription dates to the 1990s [26]

3.1 State of the art in AMT

There has been substantial progress made in the field of AMT. Neural networks, in particular, have met and surpassed the performance of traditional pitch recognition techniques on polyphonic audio.

A model used in [27] uses 87 Support Vector Machine (SVM) classifiers to perform frame-level classification with the advantage of simplicity, and then a Hidden Markov Model (HMM) post-processing was adopted to smooth the results. On top of it, Deep Belief Network(DBN) was added to learn higher layer representation of features in [28]. Since none of the approaches has reached the same level of accuracy as human experts, most music transcription work is completed by musicians.

The first major AMT work is Smaragdis et al.[29]. This approach uses Non-Negative Matrix Factorization (NMF). This is the main methodology employed in software for automatic transcription, but it has it's limitations. For example, it needs to know how many individual notes are desired for the transcription (information that is not always available).

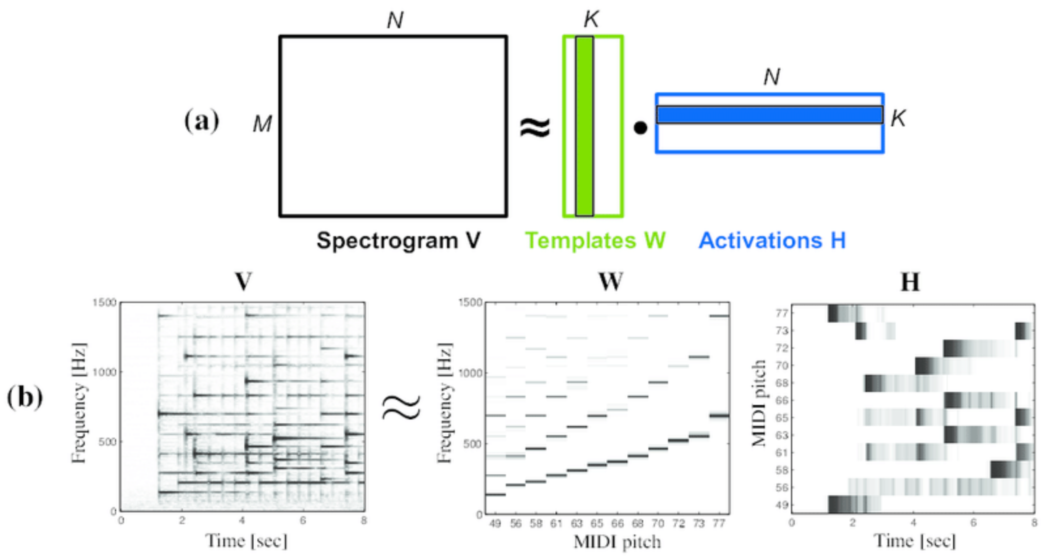
The next work worth mentioning is Emiya et al.[30], not because of their transcription system (as it was out-performed in the same year), but because of the dataset they created that has become the standard in evaluating any multi-pitch estimation system. They created the MIDI-Aligned Piano Sounds (MAPS) data set composed of around 10,000 piano sounds either recorded by using an upright Disklavier piano or generated by several virtual piano software products based on sampled sounds. The dataset consists of audio and corresponding annotations for isolated sounds, chords, and complete pieces of piano music.

Melodyne is a popular plugin used for Music Transcription and Pitch Correction. It costs up to $700. The Melodic and Polyphonic algorithms offer you, in the case of vocals as well as both mono- and polyphonic instruments, full access to the notes of which the sound is composed as well as to their musical parameters. There's no public information about what approach they used.

3.2 NMF approach to AMT

The work of Smaragdis et al.[29] was the first major AMT work. This approach used NMF. It works by factoring a non-negative matrix $\mathbf{X} \in R^{\geq 0, MxN}$ into two factor matrices: $\mathbf{H} \in R^{\geq 0, MxR}$ and $\mathbf{W} \in R^{\geq 0, RxN}$, where R is chosen, in the case of a piano roll it is 88. **H** represents when each component is active, known as the Activation Matrix and **W** shows the frequency spectrum of what should only be a single note, known as Basis Matrix. [31]

Figure 8: Non-negative matrix factorization [32]



While this is the main technique used in software for AMT, it has it's limitations. It needs to know how many individual notes are desired for the transcription, something which is not always available.

3.3 CNN approach to AMT

With the development of deep learning in recent years, researchers were inspired to apply neural networks to achieve AMT. Many models were proposed, including CNNs, Recurrent neural networks (RNN), Long-short term memory networks (LSTM) [26] [33]. Five models were compared and CNNS were reported to have the best performance. [1]

Sigtia et al.[34] built the first AMT system using CNN, outperforming the state of the art approaches using NMF. Convolutional Neural Networks are a discriminative approach to AMT, which has been found to be a viable alternative to spectrogram factorization techniques. Discriminative approaches aim to directly classify features extracted from frames of audio to the output pitches. This approach uses complex classifiers that are trained using large amounts of training data to capture the variability in the inputs, instead of constructing an instrument specific model.

Table 1: Optimal parameters and architecture for ConvNet in Sigtia et al [1]

Parameter Name	Value
Window Size (Spectrogram Frames)	7
Number of ConvNet Layers (Conv+tanh+Pooling)	2
Number of Fully Connected Layers	2
Window Shapes(1,2)	(5,25),(3,5)
Pooling Size	(1,3)
Fully Connected Widths (1,2)	1000,200

Sigtia et al. [34] explored various models for pitch detection. In addition to CNNs, they tried Deep Neural Networks and Recurrent Neural Networks. The results have shown that their CNN based model outperformed the others for this task. In their paper, they propose a Music Language Model (MLM) that's based on RNNs in order to handle the polyphonic musical data. [35]

There are some products on the market, AnthemScore for example, that use CNNs for AMT. They approach note detection as an image recognition problem by creating spectrograms of the audio. They show how the spectrum or frequency content changes over time. The method used for creating the spectrograms is the constant Q transform instead of the more common Short Time Fourier Transform (STFT) method.

4 Theia7

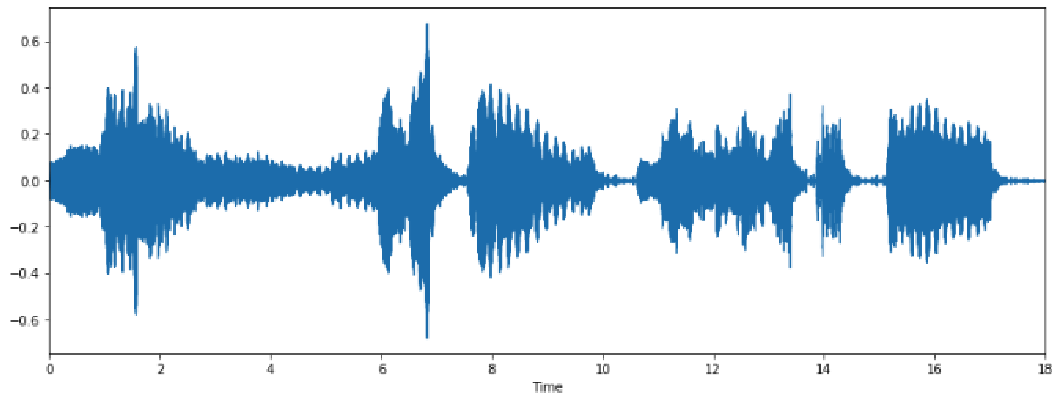
4.1 Input Representation

The data set used was the MAPS database [36]. It is a piano database for multi pitch estimation and automatic transcription of music. It contains MIDI-annotated piano recordings, composed of isolated notes, random-pitch chords, usual musical chords and pieces of music. It provides a diverse range of sounds from various recording conditions.

The recordings are CD quality (16-bit, $44 - kHz$ sampled stereo audio) and the related aligned MIDI files contain the ground truth. The overall size of the database is 40GB.

Working with sound in neural networks is different from dealing with images. The input contains audio files so before feeding it to the network we need to turn it into a visual representation. The most common way to represent a sound is the audio representation in the time domain. (Figure 9)

Figure 9: Example of audio representation in the time domain [37]



The figure 9 shows the evolution in time of the song and you can see the oscillation of the signal. the Y axis represents the amplitude while the X is the time. In this representation it's not possible to distinguish the notes that are playing. Because of this we need a better representation. This is where the Fourier Transformations come in handy. Using the transformation on the data we get a representation over frequency instead of time. It is also known as a spectrum. The spectrum reveals relevant information that's crucial to analyse the audio. (Figure 6)

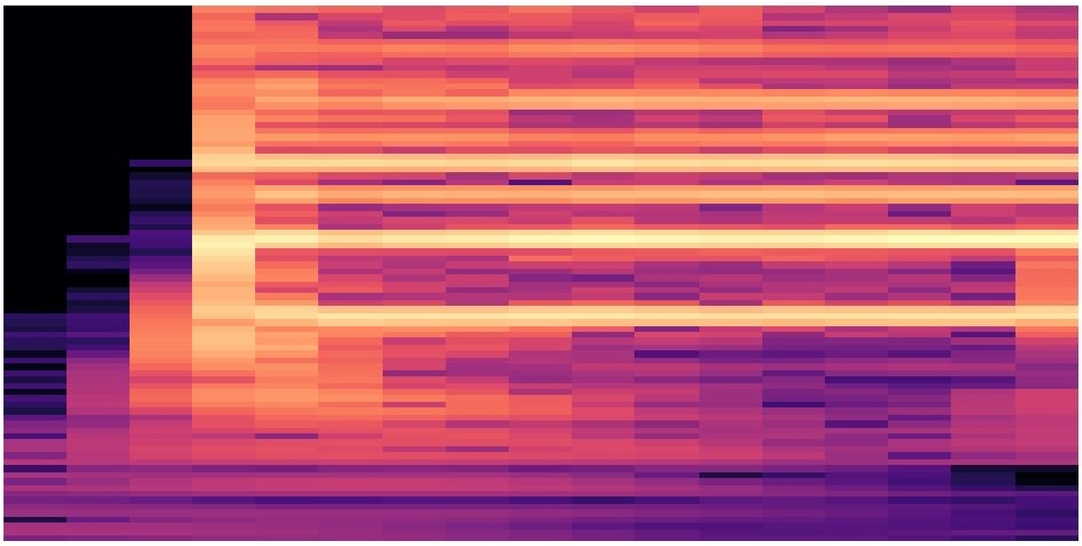
The input MIDI file will be split into $\frac{1}{16} \cdot second$ window frames. For example a note lasting 1 second will have 16 consecutive windows created. Each window is then turned into a wav file , using fluidsynth[38], which in turn will transformed using the Constant-Q transform.

To compute the Constant-Q transform, the library *Librosa* [39] will be used, in particular the method called *librosa.cqt*. The parameters that we will use are:

- **y**: Audio signal
- **sr**: Sampling rate
- **fmin**: Minimum frequency
- **n_bins**: Number of frequency bins
- **bins_per_octave**: Number of bins per octave
- **hop_length**: Number of samples between successive CQT columns

The result of the Librosa function will be plotted, resulting into a logarithmic scale spectrogram of size $145x49$ (Figure 10). This way from a song of length x, $x \cdot 16$ spectrograms will be extracted. These are the input of the network.

Figure 10: Example of a constant-q spectrogram



4.2 Labeling

A supervised machine learning model needs the data to be labeled. As it was explained in the subsection above, the input song is going to be represented in the frequency domain. The labeling will be provided using the MIDI files associated

Table 2: Example of a one hot representation of a midi file

Note	Window 1	Window 2	...	Window n
1	0	0	...	0
2	1	0	...	0
3	0	0	...	1
...	...	...	...	...
128	0	1	...	0

with the input *.wav* files. They contain information about the notes playing at a certain time. The labels will be arrays showing the played notes represented with a *one-hot encoding.*

One-hot encoding is a widely used technique in neural networks. It consists of creating an array of boolean values. [40] In this case, every column symbolizes a possible note that can be played in a certain moment. Because of this there are as many columns as possible notes (128). This will be done per window, so the goal is to see which notes are played in a certain window (Table 2). A value of 1 in a cell means that the specific musical note has been played during that window, while a value of 0 is the opposite.

4.3 First Cnn Architecture

For the first neural network has been created with the aim of obtaining some initial results and it has served to learn how a model developed in Keras [41] behaves.

An architecture similar to the Keras MNIST is evaluated[**?**]. It has been created to predict handwriting digits, the input and output data are completely different, but this is a standard architecture for image recognition with small patches. No optimization was done on this one as its aim was to perform a first approach that deals with the input data.

The architecture is shown in Table 3. It performed poorly but it helped in the preprocessing progress as it learned really fast on a GPU. Making changes to the input sizes and different types of transformations and retesting was really quick.

4.4 Proposed Cnn Architecture

The proposed architecture is similar to the AlexNet [42] (Figure 13). This architecture won Image Classification Challenge in 2012. It was used on high-resolution images into 1000 different classes. This was too big for our input data of $145x49x3$ size images. The kernel and stride sizes were adjusted to fit our

Table 3: Initial cnn architecture

Layer (type)	Output Shape
Conv2D	(None, 72, 24, 32)
Conv2D	(None, 70, 20, 64)
MaxPooling2	(None, 35, 10, 64)
Dropout(0.25)	(None, 64)
Flatten	(None, 2048)
Dense	(None, 128)
Dense	(None, 128)

data. Some layers were dropped and some dropout layers were added in order to reduce overfitting. The dropout layers have a 0.5 chance to deactivate a neuron. This forces the layer to learn the same concept with different neurons, improving generalization.

Figure 11: Accuracy for the final architecture

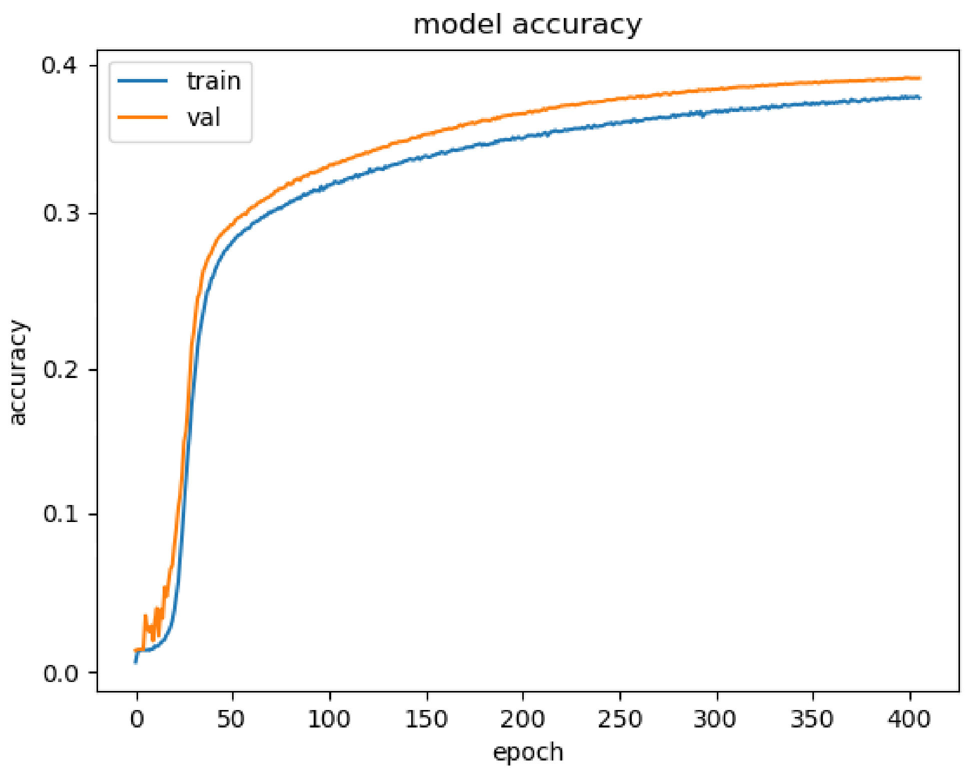
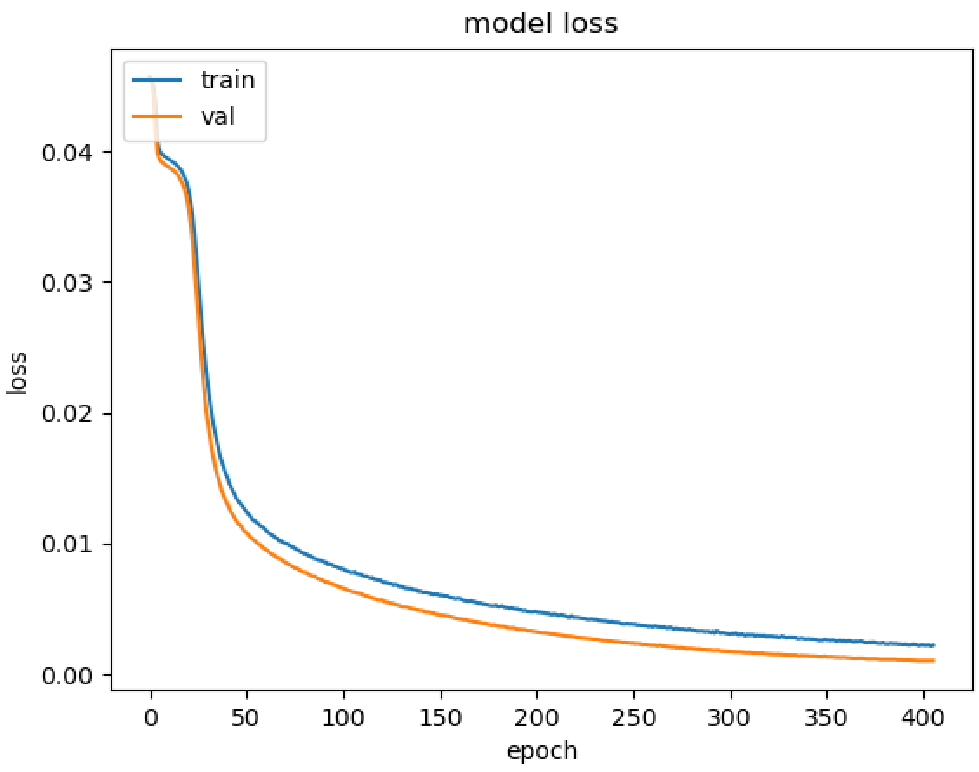


Figure 12: Loss for the final architecture



Three activation functions were tested in the hidden layers:

- **Sigmoid**: 28.83% accuracy
- **Tanh**: 33.16% accuracy
- **ReLu**: 38.73% accuracy

The model performed better using ReLu as the activation function inside the hidden layers. As for the output layer Softmax was used as this is a multi class classification problem.

Figure 13: Original AlexNet architecture [42]

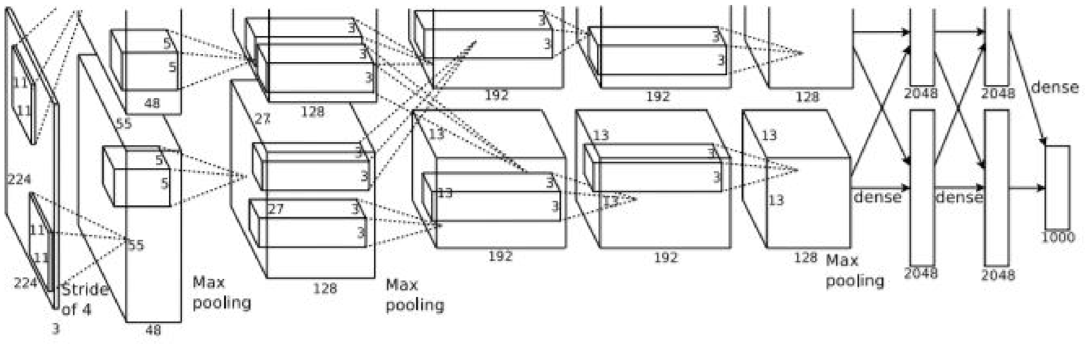
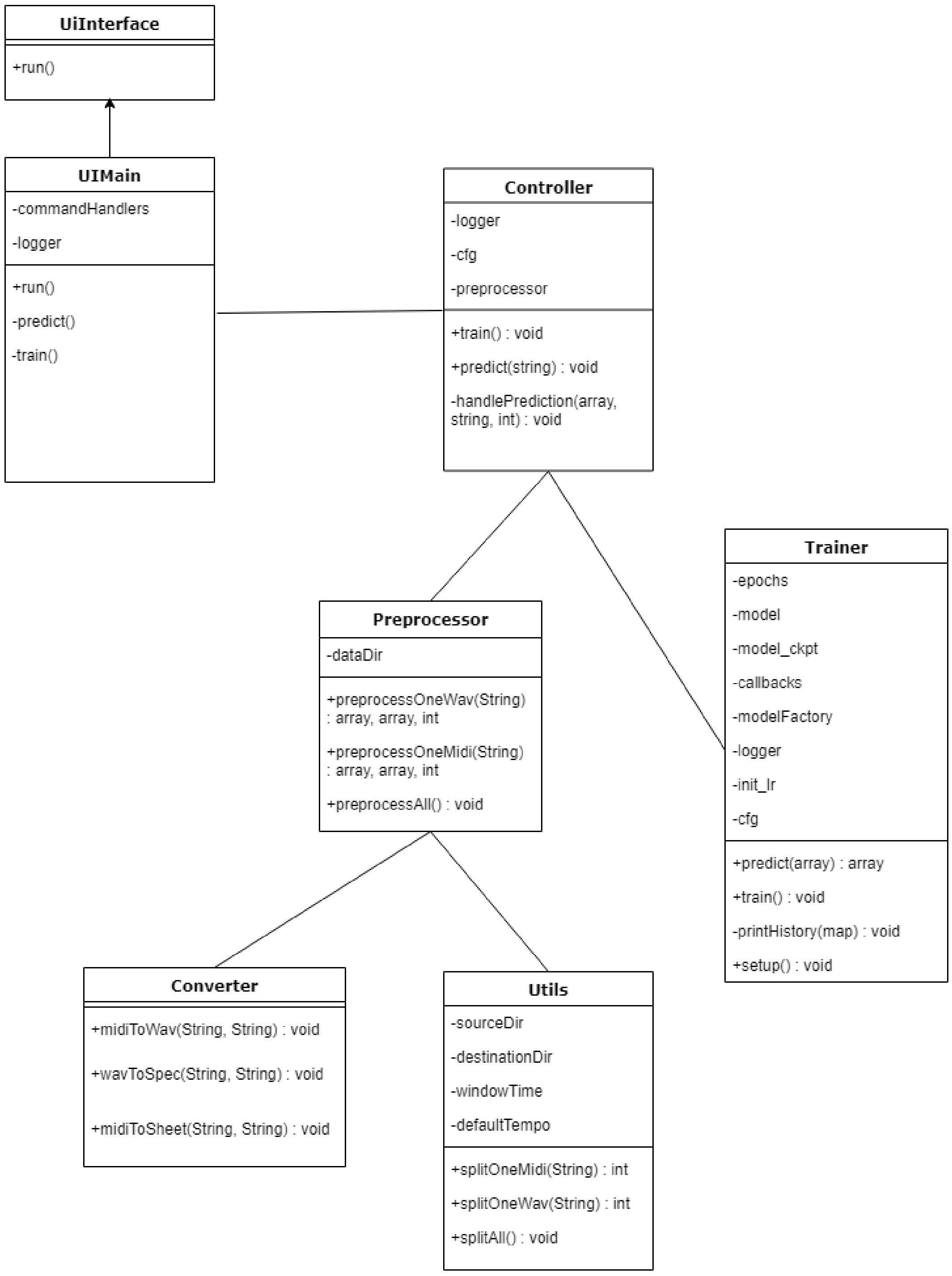


Table 4: Proposed Cnn Architecture

Layer (type)	Output Shape	Param #
Conv2D	(None, 72, 24, 32)	896
MaxPooling2	(None, 36, 12, 32)	0
Conv2D	(None, 17, 5, 64)	18496
Conv2D	(None, 16, 4, 128)	32896
MaxPooling2	(None, 8, 2, 128)	0
Flatten	(None, 2048)	0
Dense	(None, 4096)	8392704
Dropout	(None, 4096)	0
Dense	(None, 4096)	16781312
Dropout	(None, 4096)	0
Dense	(None, 128)	524416

4.5 Application architecture

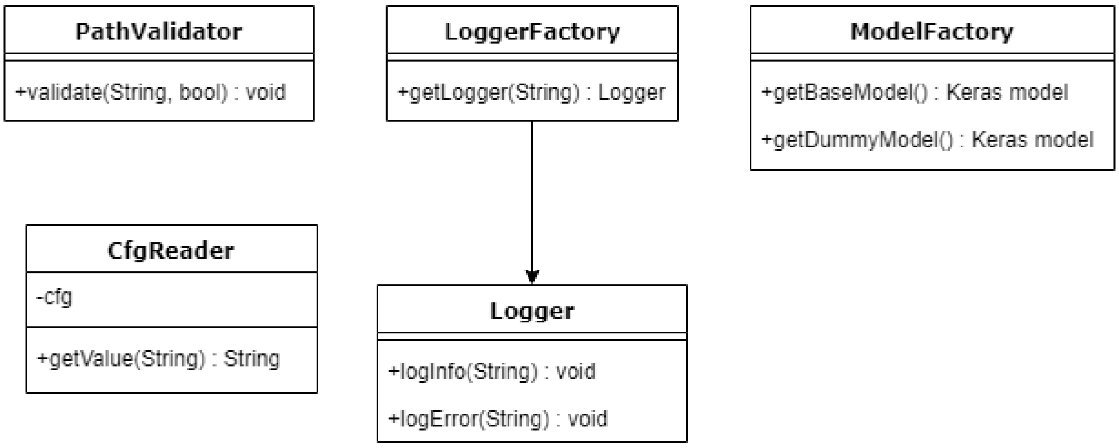
Figure 14: Application class diagram



Theia7 has a layered application without the data layer. For the model and loggers creation the abstract factory design pattern was used in order to facilitate an easy creation and scaling for these features. The main business logic happens in the controller with the trainer and preprocessor at its disposal.

The configuration of the application is contained inside a json configuration file. A wrapper over it was created to facilitate reading fields from it inside the classes 15

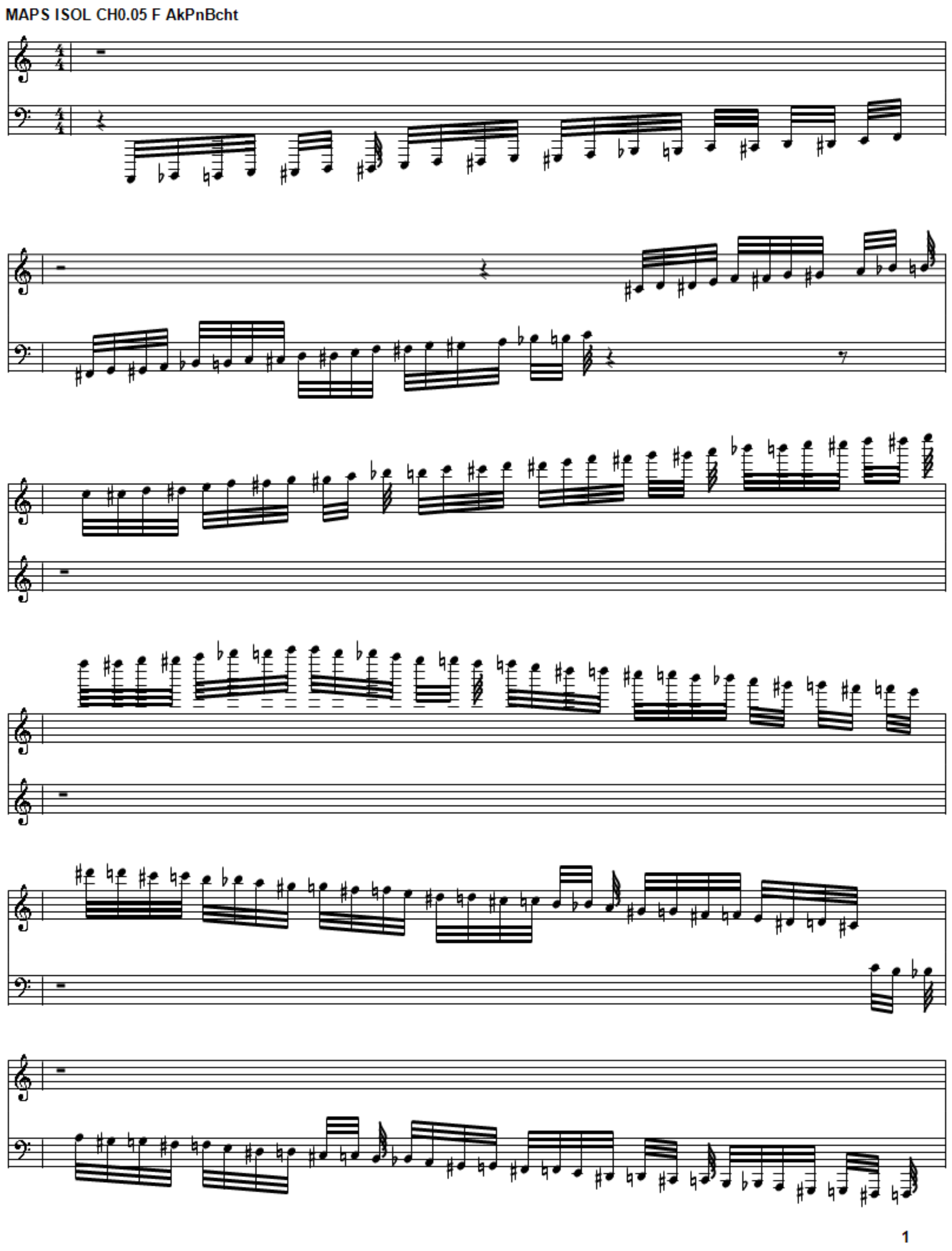
Figure 15: Static classes diagram


PathValidator
+validate(String, bool) : void
LoggerFactory
+getLogger(String) : Logger
ModelFactory
+getBaseModel() : Keras model
+getDummyModel() : Keras model
CfgReader
-cfg
+getValue(String) : String
Logger
+logInfo(String) : void
+logError(String) : void

4.6 Output

The output of the prediction will be a one-hot encoding of the song as described in Table 2. This will be transformed into a midi file using the pretty midi library [43]. Knowing that each window is a $\frac{1}{16}$ of a second we can find an approximation of the original note length by concatenating all consecutive windows predicted for it. The midi is then converted to wav using fluidsynth [38] to have a sound comparison and the spreadsheet is created using the Sheet software [44]. An output example can be found in Figure 16.

Figure 16: Sheet output example



4.7 Technologies used

List of used technologies:

- Librosa [39]
- Keras [41]
- Tensorflow [45]
- Pretty-midi [43]
- Sheet [44]
- Numpy [46]
- Pydub [47]
- Mido [48]
- Cuda [49]

5 Testing

The application was tested using unit-testing for various components:

- Path validators were given non-existing, existing paths, directories when testing for normal files and vice versa.
- Preprocessing utils were tested to see if the songs are split into the correct number of windows
- One-hot encoding and decoding utils

  Code snippets will be added here

6 Conclusion

6.1 Overview

The recognition of musical notes played in a song is a complex problem to deal with because of the polyphonic aspect of music and it's still an unsolved problem. The neural network has to be able to distinguish the notes from different instruments. Because of this a Constant-Q transform was applied. This removes instruments from the equation, normalizing the data and obtains valuable information from the song. The MAPS dataset [36] provides raw labels for each song but they don't fit the expected output of the network. Because of this each midi provided needs to be analyzed and one-shot encodings of size 128 are extracted for each $\frac{1}{6}$ of a second window of the song to determine what notes are played during this time frame. The windows are then converted into wav format and then into spectrograms using the Librosa [39] library. The spectrograms are then fed to the network's learning process. The learning set is split 80% for training, 20% for testing and 10% of the training set is used for validation.

The output is a one-shot encoding of each window. These are then glued together and a midi file is created. From this midi file a wav file is exported for comparison and the music sheet is generated using the Sheet freeware. [44]

6.2 Results

The proposed model achieved a 38.73% general accuracy for the used data-set, which was a mix of polyphonic and monophonic songs. It struggled with overlapping notes sometimes predicting only the harmonics or bass notes. For the monophonic part of the problem it predicts songs with an accuracy of 99.5%.

Taking into account that the state-of-the art methods from Google Magenta are around 70% this result is pretty good, given no digital signal processing prior knowledge.

6.3 Future work

There are ways to improve the accuracy further:

- A long short-term memory network could be added to aid with the classification and detect the length of the notes
- Genre detection could also aid with the classification as it will tell if a note makes sense in a progression

- Further improvements in the preprocessing. By changing the spectrogram from a STFT to a Constant-Q the accuracy improved significantly. There are more ways the make the input clearer by reducing the noise and improve the window splitting. For this task more knowledge of digital signal processing is needed.
- Train on more data. Converting one window takes 0.3 seconds. A song is split into $16 * n$ windows, where n is the song length in seconds. More than two billion midi windows were created but only 160 thousands were actually converted into spectrograms due to time constraints.

The final goal of Theia7 is to be integrated in a mobile music teaching and music sheet database application, where people could share their ideas and music sheets with everyone. Theia would aid people with no musical theory knowledge by creating the sheets for them. This would disrupt the market as many people ask for money in exchange for their music sheets, a common practice for youtube cover channels.

References

[1] E. B. S. Sigtia and S. Dixon, *An end-to-end neural network for polyphonic piano music transcription.* IEEE/ACM Trans. Audio Speech Lang. Process., 24, 927–939, 2016.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

[3] "Neuron." `https://en.wikipedia.org/wiki/Neuron`.

[4] W. . P. McCulloch, *A logical calculus of the ideas immanent in nervous activity.* W. Bulletin of Mathematical Biophysics (1943) 5: 115, 1943. `https://doi.org/10.1007/BF02478259`.

[5] B. C. Bangal, *Automatic generation control of interconnected power systems using artificial neural network techniques.* 2009. `https://doi.org/10.1007/BF02478259`.

[6] *Everything you need to know about Neural Networks* . 2018.

[7] "Activation functions." `http://prog3.com/sbdm/blog/cyh24/article/details/50593400`.

[8] "Vitaflux." `vitalflux.com`.

[9] "Convolutional neural networks." `https://en.wikipedia.org/wiki/Convolutional_neural_network`.

[10] "cnn representation." `https://en.wikipedia.org/wiki/File:Typical\_cnn.png`.

[11] "Convolutional neural networks." `https://cs231n.github.io/convolutional-networks/`.

[12] "Digital Signal Processing." `https://www.analog.com/en/design-center/landing-pages/001/beginners-guide-to-dsp.html`.

[13] "Physics of sound." `https://method-behind-the-music.com/mechanics/physics/`.

[14] "Traveling wave." `https://method-behind-the-music.com/mechanics/physics/`.

[15] "Speed of sound." `https://en.wikipedia.org/wiki/Speed_of_sound`.

[16] G. Sahidullah, Md.; Saha, *A Novel Windowing Technique for Efficient Computation of MFCC for Speaker Recognition.* IEEE Signal Processing Letters. 20 (2): 149–152, 2013.

[17] "Short time fourier transform example." `https://en.wikipedia.org/wiki\/short-time\_fourier\_transform`.

[18] "Constant-Q transform." `https://en.wikipedia.org/wiki/Constant-Q_transform`.

[19] Z. Duan and E. Benetos, *Automatic Music Transcription.* 2015.

[20] "Music information retrieval evaluation exchange.." `http://www.music-ir.org/mirex/wiki/2017:Multiple_Fundamental_Frequency_Estimation_\%26_Tracking_Results_-_MIREX_Dataset`.

[21] "Transcription." `https://en.wikipedia.org/wiki/Transcription_(music)`.

[22] E. S. D. H. A. Klapuri, *Automatic music transcription: Breaking the glass ceiling.* 2012.

[23] A. Livshin and X. Rodet., *Musical instrument identification in continuous recordings.* 2004.

[24] "Sound waves of different musical instruments." `http://eyuzdaosivadare.changeip.com/Sound-waves-and-musical-instruments.html`.

[25] H. K. S. Dixon and A. Klapuri, *Multi-template shift-variant non-negative matrix deconvolution for semi-automatic music transcription.* 2012.

[26] D. G. Morin, *Deep neural networks for piano music transcription.* 2017.

[27] G. E. Poliner and D. P. Ellis, *A discriminative model for polyphonic piano transcription.* EURASIP Journal on Applied Signal Processing, vol. 2007, no. 1, pp. 154, 2007.

[28] J. N. J. Nam, H. Lee, and M. Slaney, *A classification-based polyphonic piano transcription approach using learned feature representations.* " Proceedings of the 12th International Society for Music Information Retrieval Conference, pp. 16-180, 2011.

[29] P. Smaragdis and J. C. Brown., *Non-negative matrix factorization for polyphonic music transcription.* In Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on., pages 177–180. IEEE,, 2003.

[30] R. B. V. Emiya and B. 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, 2010.

[31] "Non-negative matrix factorization." `https://en.wikipedia.org/wiki/Non-negative_matrix_factorization`.

[32] S. Ewert and M. Muller, *Score-informed source sepa-ration for music signals.* 2012.

[33] J. F. S. B. L. Sturm, O. Ben-Tal, and I. Korsunova, *Music transcription modelling and composition using deep learning.* arXiv preprint arXiv:1604.08723, 2016.

[34] J. Sleep, *AUTOMATIC MUSIC TRANSCRIPTION WITH CONVOLUTIONAL NEURAL NETWORKS USING INTUITIVE FILTER SHAPES.* A Thesis presented to the Faculty of California Polytechnic State University, San Luis Obispo, 2017.

[35] S. D. E. Benetos, *A shift-invariant latent variable model for automatic music transcription.* 2013.

[36] V. E. R. Badeau and B. David, *Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle.* 2008.

[37] "Music genre classification." `https://towardsdatascience.com/music-genre-classification-with-python-c714d032f0d8`.

[38] "Fluidsynth." `http://www.fluidsynth.org/`.

[39] "Librosa library." `https://librosa.github.io/librosa/`.

[40] "One-hot encoding." `https://en.wikipedia.org/wiki/One-hot`.

[41] "Keras." `https://keras.io/`.

[42] A. K. I. Sutskever and G. E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks.* 2012.

[43] "Pretty midi." `https://github.com/craffel/pretty-midi`.

[44] "Midi to sheet converter." `https://github.com/BYVoid/MidiToSheetMusic`.

[45] "Tensorflow." `https://www.tensorflow.org/`.

[46] "Numpy." `https://www.numpy.org/`.

[47] "Pydub." `http://pydub.com/`.

[48] "Mido." `https://mido.readthedocs.io/en/latest/`.

[49] "Cuda." `https://developer.nvidia.com/about-cuda`.