Music notation generation

DAT5550 Data Mining 2021 - Final Project Report

Asahi Cantu Moreno
University of Stavanger, Norway
a.cantumoreno@stud.uis.no

Tim Bauerle
University of Stavanger, Norway
t.bauerle@stud.uis.no

ABSTRACT

Creating music notation, also called music transcription is in very tedious and and time consuming process both for amateur and professional musicians. Hence, it is desirable to automate it and deep learning algorithms helped to get closer to achieving this. By using state of the art technologies, we developed a case study to try different models (for Machine Learning and Deep Learning) to transcript music audio files into musical notation. This project involved the analysis of several piano pieces in MIDI[1] format converted into Wave files, with subsequent data processing using advanced audio analysis techniques to extract frequency-spectrograms and derivatives. With this data tuned, refined and normalized the following different approaches were tried:

(1) Multilayer perceptrons
(2) Long Short Term Memory Neural Networks (LSTMs)
(3) Convolutional Neural Networks (CNNs)

Finally, the results for each model based on different input features are presented and the best performing model for music transcription is determined. The objective of this project was to generate a program able to receive piano music audio file as an input and generate its musical notation as accurate as possible.

KEYWORDS

Music Notation, Automatic Music Transcription, Data Mining, Deep Learning, Machine Learning, MIDI, music, musical notation, Audio Analysis, Spectrogram, CNN, LSTM, Feature Extraction, Image recognition

0.1 Contributions

Contributions and participation over the development of the project and report are given as follows:

- Exploratory Data Analysis and preprocessing for audio music: Asahi, Tim
- Feature extraction and audio Analysis: Asahi, Tim
- Multilayer perceptron, LSTM: Tim
- Convolutional Neural Networks: Asahi
- Project architecture: Asahi, Tim
- Report redaction: Asahi, Tim

[1]Musical Instrument Digital Instrument. Protocol for music [11]

Supervised by Vinay Setty .

1 INTRODUCTION

With the progress of Deep Learning algorithms, sciences and arts have been benefiting by many clever models and approaches pretending to solve different problems in the music industry and audio research such as speech recognition, audio classification, sound classification and identification. Audio and sound extraction, analysis and research are intriguing subjects for the Deep learning exploration, application and creation due to the unstructured nature of the data. Nowadays music analysis and generation are being researched and explored through advanced computer models capable of "listening" and distinguishing noise from voices, melodies, different types of sounds and even compose music. The music industry in the age of digital distribution craves form smart methods, models and approaches that allow to create new ways of listening, exploring and provide features to users and musicians that allows companies better market capture and play a leadership and role model for its business in the near future. As an example the some of the most innovative projects in the field are:

- Speech to text [4]
- Text to speech [3]
- Automatic voice generation[12]
- Music composition [8]
- Animal species recognition [9]
- Musical Instrument classification [7]
- Audio Genre classification [10]
- Automatic Music Transcription [15]

Music transcription was very interesting research subject to the team, and therefore the field of study for this report. Automatic music transcription [2] is considered a hard and still open problem in music processing [5]. There is not an ideal nor best solution found yet to an algorithm or model that can listen and transcribe any kind of music. As the technology in Computer Science and hardware capabilities progress, new possibilities and fields of exploration unleash, with a potential to discover and create drastically new solutions for this field.

The factors that revolutionized music for this case study are:

- Massive Digitization of music (P2P Network, Music Streaming).
- Creation and implementation of a universal language (MIDI) to allow musical instrument with computer communication.
- Internet velocity increase
- Hardware and computational resources increase
- Popularization of Python[3] and creation for multiple libraries for audio analysis, processing, machine learning algorithms and frameworks.

[2](AMT), i.e. the process of detecting notes and generating a (possibly human readable) notation
[3]Programming language

Name	#Files	Size MB	Avg Duration (min)
Classical Music MIDI[13]	50	2.4	6

Table 1: Dataset specifications

- Implementation of cloud computing and lots of free resources to test and code over big datasets
- Literature and previous research papers that had already explored this field.

With these elements accessible as tools and resources it was possible to:

(1) Explore and download a candidate data set of MIDI Files
(2) Use Python language to extract desirable MIDI Features and compare among feature extraction techniques
(3) Use Python language to convert MIDI files into Wave (.Wav) Files
(4) Extract features through advanced audio analysis techniques
(5) Explore different Deep Learning algorithms and develop suitable models
(6) Train the models and compare performance
(7) Keep a candidate model and generate the notation for a piece of music.

2 DATA PRE-PROCESSING

To properly understand the workflow performed to process the dataset into suitable values to train the Machine Learning models, advanced audio analysis techniques were employed to extract useful information from the wave files.

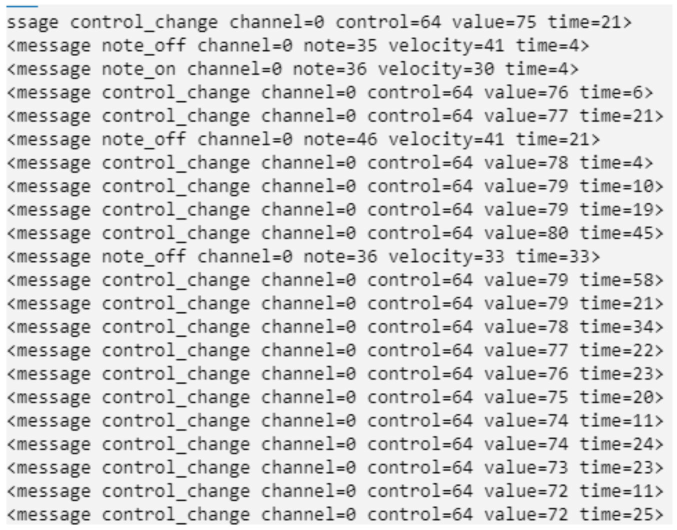
2.1 Dataset Specifications

One dataset was used for this project, which is specified in table 1. It is publicly available and continuously updated. The dataset consists of several MIDI files, small in size, but substantial for the set of instruction each one of the file counts to unleash melodies. Unlike other audio format, MIDI files contain a set of instructions to play specific notes under certain time tempo, pitch and velocity, see figure 1, which in conjunction generate a melody with one or more instruments previously selected. Each Instrument is played in one of the possible 16 channels. For research purposes only polyphonic melodies with one instrument were selected.

Once the notes are sent to a MIDI player, they are processed via a synthesizer, which contain a set of audio algorithms or previously recorded audio samples equivalent to the note to play (with its specific properties), and generates an analog signal equivalent to the chosen instrument to be played. For this process a sound font is used to produce the sounds.

2.2 Spectrogram

A spectrogram can be understood as a three dimensional feature chart represented by the time (X axis) and the frequency in a specific time frame (y axis). The strength of the wave is represented by a light color where it highlights the Decibel-intensity of a specific frequency. Thus, an audio wave can be transformed and visualized by the spectrum of frequencies varying with the time. The spectrogram is useful for audio analysis since it represents a picture in time for a sound and therefore a specific signature from it. Derivative and advanced models of spectrograms were used as well to extract audio features.



```
ssage control_change channel=0 control=64 value=75 time=21>
<message note_off channel=0 note=35 velocity=41 time=4>
<message note_on channel=0 note=36 velocity=30 time=4>
<message control_change channel=0 control=64 value=76 time=6>
<message control_change channel=0 control=64 value=77 time=21>
<message note_off channel=0 note=46 velocity=41 time=21>
<message control_change channel=0 control=64 value=78 time=4>
<message control_change channel=0 control=64 value=79 time=10>
<message control_change channel=0 control=64 value=79 time=19>
<message control_change channel=0 control=64 value=80 time=45>
<message note_off channel=0 note=36 velocity=33 time=33>
<message control_change channel=0 control=64 value=79 time=58>
<message control_change channel=0 control=64 value=79 time=21>
<message control_change channel=0 control=64 value=78 time=34>
<message control_change channel=0 control=64 value=77 time=22>
<message control_change channel=0 control=64 value=76 time=23>
<message control_change channel=0 control=64 value=75 time=20>
<message control_change channel=0 control=64 value=74 time=11>
<message control_change channel=0 control=64 value=74 time=24>
<message control_change channel=0 control=64 value=73 time=23>
<message control_change channel=0 control=64 value=72 time=11>
<message control_change channel=0 control=64 value=72 time=25>
```

Figure 1: MIDI messages extracted from a sample MIDI file.

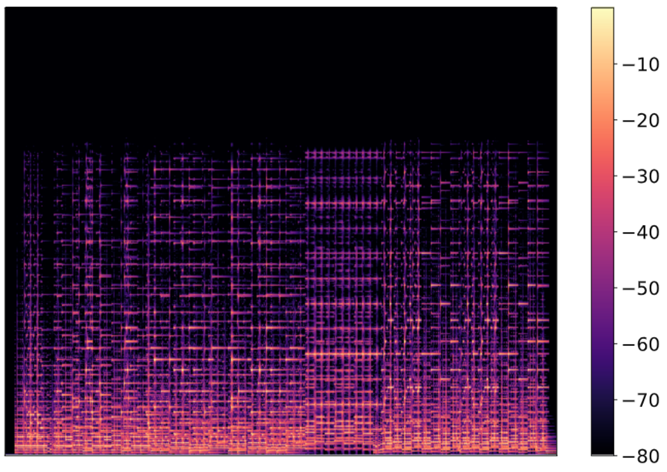


Figure 2: Spectrogram from a sample Wave file.

2.3 Mel Spectrogram

A Mel spectrogram or Mel-scaled spectrogram shown in figure 3 is a spectrogram where the frequencies are converted to the Mel-scale. In the Mel-scale, the steps are scaled in a way such that equal distances two steps sound equally distant to the listener. In particular, the frequency difference between steps increases as the frequencies get higher. This was created to give more relevance to low and mid frequencies compared to higher frequencies.

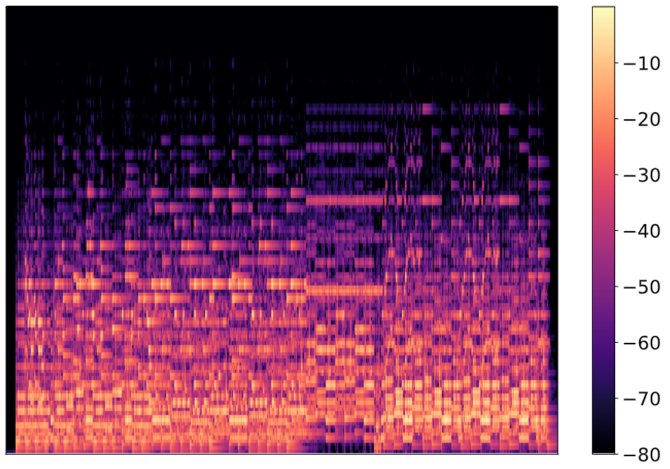


Figure 3: Mel-scaled spectrogram

2.4 Mel-Frequency Cepstral Coefficients (MFCCs)

Mel-Frequency Cepstral Coefficients (MFCCs) represent a sound short-term power spectrum based on a linear cosine transform of a log power spectrum on a nonlinear Mel-scale of frequency. A set of cepstral coefficients thus makes up the MFCC. An example of an MFCC spectrogram is shown in figure 4. As the name suggests it also uses the Mel-scale to create a higher contrast for a sound representation.

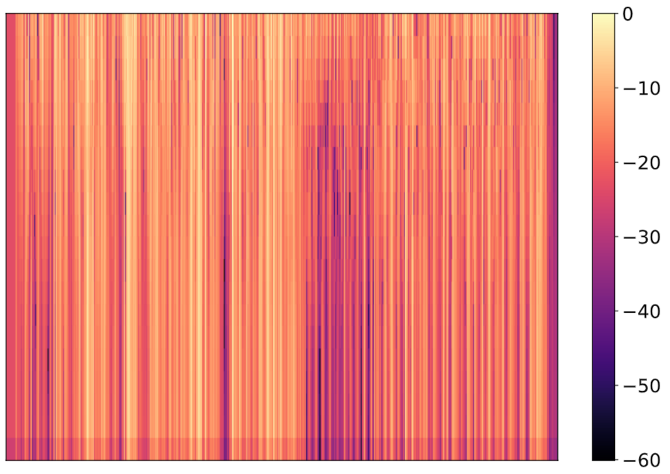


Figure 4: MFCC Spectrogram

2.5 Constant-Q Transform (CQT)

The Constant-Q Transform (CQT) represent a data series relative to the frequency domain spaced in frequency in a logarithmic scale. Each frequency filter in the transformation is proportionally spaced from its predecessor's width[14]. A CQT spectrogram is shown in figure 5.

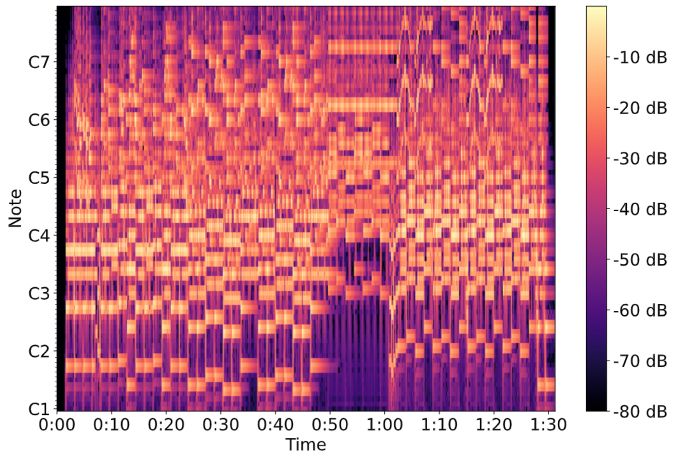


Figure 5: CQT Spectrogram

2.6 Workflow

The preprocessing workflow as depicted in figure 6 is as follows:

(1) MIDI files are read and its notes get extracted
(2) MIDI data is processed in chunks of 1/8th of a second.
(3) Resulted chunks are stored in an 88 Vector length collection. This is because a standard keyboard has 88 notes, although MIDI supports more notes. Any note in the MIDI files outside the keyboard range will be disregarded (see7).
(4) Each MIDI file is then synthesized to a wave file using a piano sound font. Selected sound font is **198-Yamaha-SY1-piano.sf2**[4]
(5) Obtained Wave files are then decomposed and processed to extract Mel, MFCC and CQT spectrograms as feature vectors.
(6) Obtained vectors are also normalized by equalizing all amplitude values.
(7) All those feature vectors are sliced as well into 1/8th of a second chunks. Where each segment will correspond to one MIDI chunk[5].
(8) Finally, all those chunks are also converted into spectrum images to future Machine Learning Training

3 EXPLORATORY DATA ANALYSIS

Data is analysed in form of spectrograms and labels after once the preprocessing state is completed since raw MIDI or wave files do not provide much insights for this case study unless explicitly transformed into usable features. The label vectors have 88 entries, one for each key, and since not many keys are played at the same time, the vector consists mostly of zeros. Overall, a key is played on average only 2.23% of the time, in other words 2.23 keys active in each 1/16th time chunk. Figure 8 shows the frequency of each key being played. The keys in the middle of the keyboard are played most frequently while the marginal keys are played less.

[4]The Sound font standard, developed by Emu Systems and their parent company, Creative Labs, is a data format that contains the detailed information necessary to create musical notes or sound effects using wavetable synthesis technology

[5]1/8 was a standard scale used since some of the songs will play at higher tempos. We assume that this small chunk will contain atomic data

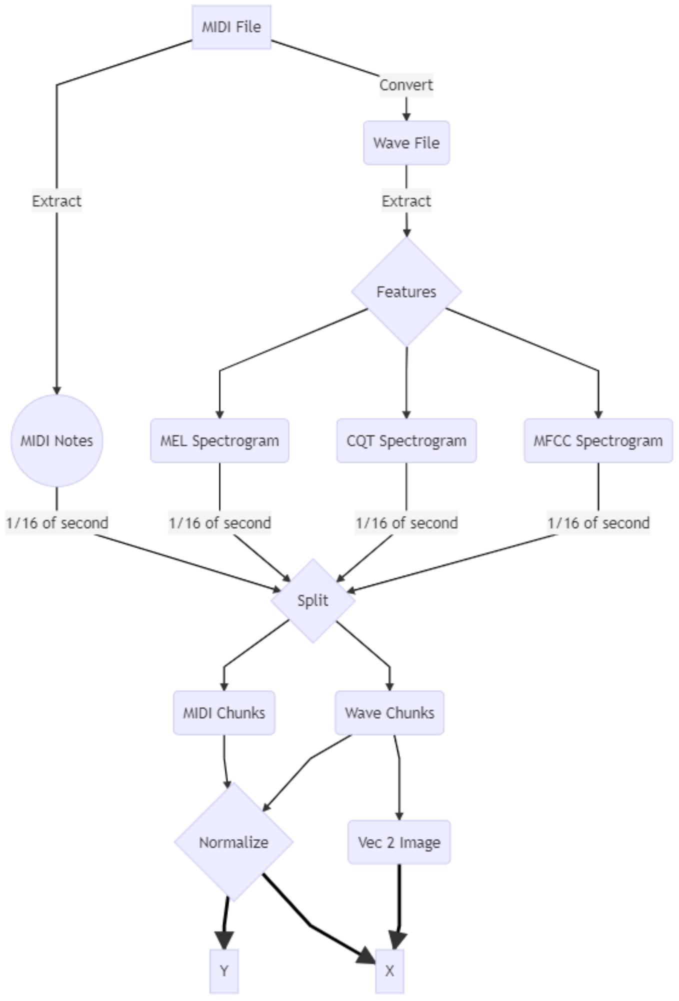


Figure 6: EDA Worflow

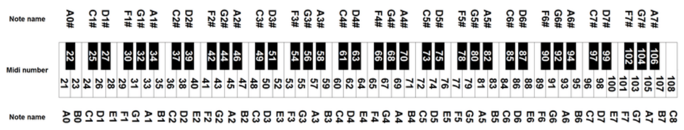


Figure 7: MIDI 88-Note Standard

In the following, we analyse the different spectrogam data. First of all, we did a PCA to see if dimensionality reduction could be useful. Therefore we computed all 800 principal components for each spectrogram type and determined the explained variance. As a result, we need the first 599 components of the Mel spectrograms to explain 99.9% of the variance. For the MFCCs we need the first 780 and for the constant-Q transform the first 621 components to explain 99.9% of the variance. Because the dimensionality cannot be decreased drastically this way and timing information is lost in the process, we decided not to use PCA to preprocess the inputs for the neural networks.

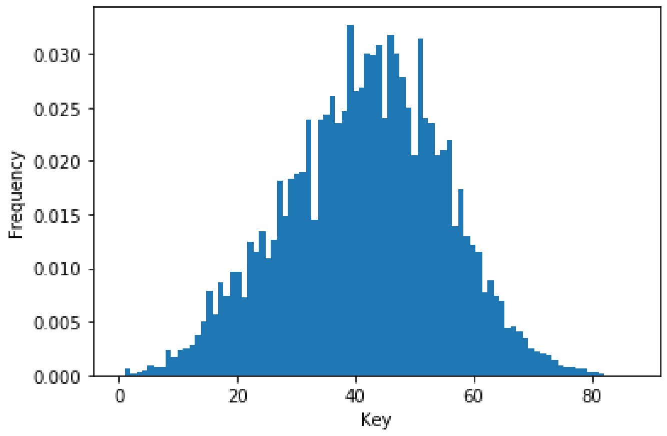


Figure 8: Relative key frequencies in the dataset

In the next step, we used PCA and TSNE for plotting the data. First, we reduced the input dimensionality to 100 by PCA and applied TSNE afterwards to reduce to 2 dimensions. The result is shown in figure 9, where the blue points correspond to Mel spectrograms, the red points to MFCCs and the yellow ones to CQT spectrum. Apparently both the scale and the shape is very similar for all types of data. Thus, we assume that similarities and dissimilarities are preserved by the different feature extraction methods.

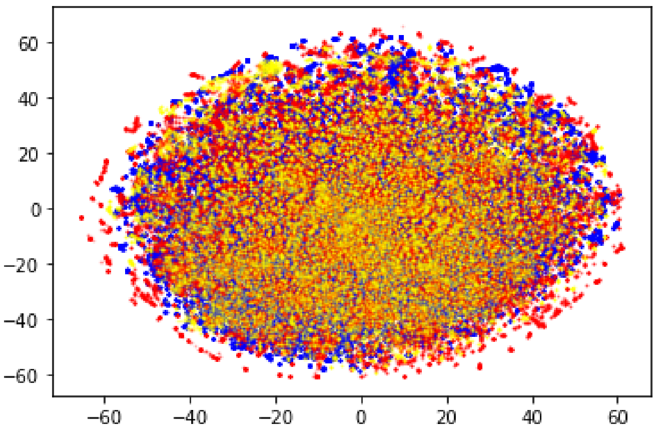


Figure 9: TSNE visualization of the dataset

4 ALGORITHMS AND WORKFLOWS

In the following, the applied models are described. Each model was trained with each type of input features. As an optimizer we used 'rmsprop' and as a loss function binary crossentropy. In section 5 we discuss the choice of loss function for our purposes.

4.1 Multilayer Perceptron

As a baseline approach, we chose a multilayer perceptron with one hidden layer. The spectrogram data is flattened before being input

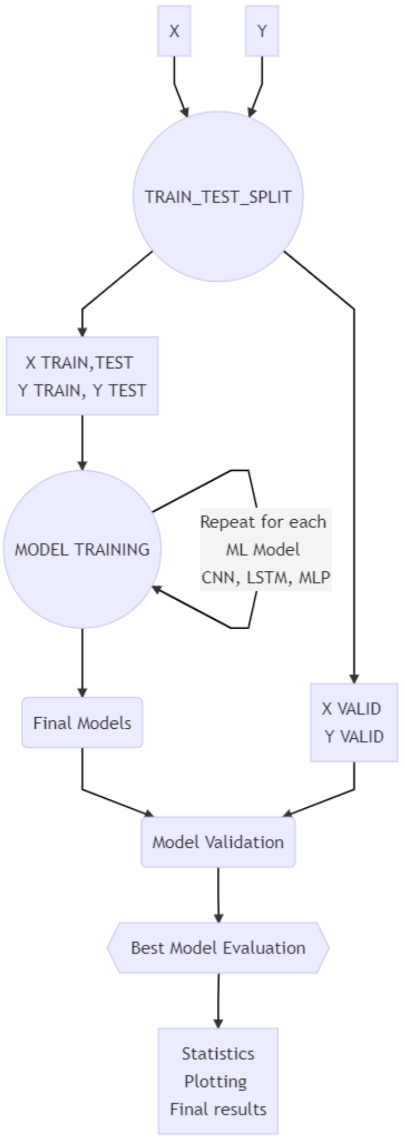


Figure 10: Model Training worflow

to the network which means it is transformed from 2D to 1D. For the hidden layer we use 512 neurons with 'relu' as an activation function and a dropout rate of 20%. The output layer consists of 88 neurons using the 'sigmoid' activation function. Also we trained a multilayer perceptron with three hidden layers of 512 neurons with 'relu' activation function and dropout rate of 20% for each layer.

4.2 Convolutional Neural Network

Furthermore we decided to do CNNs as an interesting case study by using an image representation for a specific time-frame spectrogram (for each CQT, MFCC and MEL spectrogram). Once each image is processed, a size reduction was done over several iterations to find the most suitable size at which the different CNN architectures were able to capture the most of their features when being trained.

4.2.1 CNN Architectures. There is not a definitive CNN architecture to be adopted as a standard approach, therefore different Architectures were used:

- ALEXNET NN has demonstrated to outperform over several other CNN approaches [1]
- MNIST CNN extracted from [2]
- Custom CNN build based on the first and second selections after several tuning iterations.

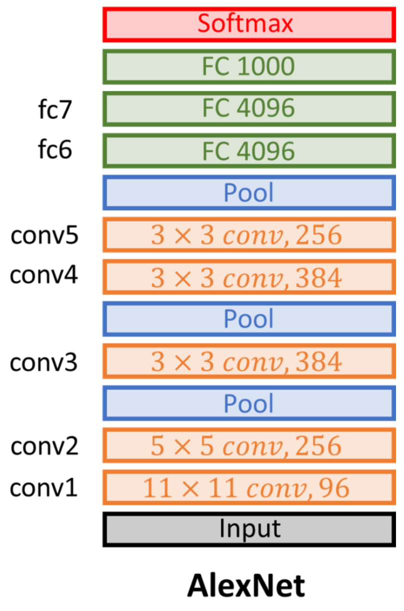


Figure 11: ALEXNET Architecture

4.3 LSTM

Using long short-term memory (LSTM) networks is an interesting approach for this project since LSTMs are suitable for sequential data which music certainly belongs to. We consider two different architectures regarding LSTMs, namely many-to-one and many-to-many LSTMs.

4.3.1 Many-to-one. The spectrogram data after preprocessing are three dimensional, i.e. samples as dimension zero, features as dimension one and time as dimension 2. For a many-to-one LSTM, we need to swap the feature and time dimensions, so the network processes the data timestep by timestep. The data is then processed by three LSTM layers with 200 neurons each. On top of that there are two more dense layers of 200 neurons each and the output layer with 88 neurons. So as each spectrogram is processed by its time dimension and an output is generated for each spectrogram.

4.3.2 Many-to-many. As the spectrograms are generated from music which is a sequence over time, we can also treat each spectrogram as a timestep and process a sequence of spectrograms with a sequence of labels. Therefore, the spectrograms are flattened and grouped in sequences of length ten corresponding to 1.25 seconds of music. Each group then corresponds to one sample in the data. The architecture of the network is almost the same as previously, with 3 LSTM layers of 200 neurons each and two dense layers on top of 200 and 88 neurons respectively. The only difference is that all LSTM layers output a sequence, while in the many-to-one network the last LSTM layer does not output a sequence. Due to that slight modification, the many-to-many network generates ten predicted labels for each sample.

5 CHALLENGES AND LIMITATIONS

5.1 Audio synthesis and audio analysis

Music recognition in general is a hard problem as the sound of every note consists of multiple frequencies, which mix up when multiple notes are played at the same time. Apart from the pitch there are other properties for each note as well like duration or volume being desirable to detect as well. We mainly focused on the pitch and implicitly as well on the duration, due to the processing in chunks. A major challenge is that a real piano recording of several minutes duration in a wave file format cannot be used as training input for a neural network since there is no suitable way to determine the labels. Therefore we had to use MIDI recordings which have the disadvantage to only produce a synthetic sound for the further processing. Overall, the preprocessing was a complicated and time consuming procedure, due to the number of necessary steps described in section 2.

5.2 Output Encoding and loss function

A piano has 88 keys and usually multiple keys can be pressed at the same time. Thus a one-hot encoding is not suitable due to a large number of key combinations, neither can the output be a probability distribution which is desirable for a couple of loss functions. Thus, we had to use an output between zero and one for each key individually and determine a decision threshold later to classify whether this key is played in the current chunk of music.

In general we judge the quality of the output using measures like accuracy defined on a confusion matrix as in a binary classification problem. Unfortunately these measures cannot be reflected properly by the loss function, since the threshold based classification prohibits a smooth and differentiable loss function.

5.3 Training ML models

A common issue for all model was that all predicted outputs tended to become close to zero. This is not a surprise, since the dataset is imbalanced (more labels are zero than one) and always predicting zero gives a low loss value. Using dropout seemed to be a helpful countermeasure here.

5.3.1 CNN Algorithm Adjustment and tuning. It was difficult to create a suitable CNN model in the beginning that could perform accurate evaluations. Most of the first trials failed at the second epoch. After a while and since many models were not able to handle so many different features, we decided to reduce its size but keep still RGB dimensions since it is thought it could give better pattern and shape insights when each channel convolves over the whole pixel-matrix. Several times even the machine overheated and rebooted, making the process of training tedious and exhausting.

Model	Accuracy	Precision	Recall	F1	AUC
Multilayer Perceptron (1 hidden layer)					
CQT	0.8692	0.0633	0.3336	0.1065	0.7642
MEL	0.8965	0.0691	0.2750	0.1104	0.7689
MFCC	0.8589	0.0642	0.3711	0.1094	0.7639
Multilayer Perceptron (3 hidden layers)					
CQT	0.8468	0.0589	0.3707	0.1016	0.7636
MEL	0.8645	0.0650	0.3587	0.1100	0.7738
MFCC	0.8706	0.0670	0.3515	0.1126	0.7741
LSTM (many-to-one)					
CQT	0.8379	0.0595	0.4010	0.1036	0.7024
MEL	0.8174	0.0548	0.4198	0.0970	0.6597
MFCC	0.8388	0.0594	0.3980	0.1034	0.6869
LSTM (many-to-many)					
CQT	0.8398	0.0568	0.3753	0.0986	0.7591
MEL	0.7784	0.0550	0.5245	0.0996	0.7591
MFCC	0.7532	0.0542	0.5812	0.0991	0.7585
CNN2D					
CQT	0.7882	0.0572	0.5014	0.1027	0.7610
MEL	0.7346	0.0548	0.6138	0.1006	0.7687
MFCC	0.8069	0.0567	0.4471	0.1007	0.7627
ALEXNET					
CQT	0.0242	0.0242	1.0	0.0472	0.4987
MEL	0.0242	0.0242	1.0	0.0472	0.4969
MFCC	0.0242	0.0242	1.0	0.0472	0.500
CNN2D V2					
CQT	0.7746	0.0573	0.5385	0.1035	0.7703
MEL	0.7742	0.0564	0.5306	0.102	0.7692
MFCC	0.8053	0.0587	0.4688	0.1043	0.7697

Table 2: Evaluation Results

5.4 Time and experience

The limited amount of time and experience in the development of this kind of projects is a key limitation. It is important to highlight that there will be many different approaches for this problem and maybe different classifiers could outperform the proposed one.

5.5 Computational resources

The computational resources required to train deep learning models prevented several time to finish the training processes.

6 RESULTS

The results of each model type based on the validation set are shown in table 2. We use a 60-20-20 split for training, test and validation data. For the splitting we used a fixed random state, so the splitting is always the same for all model types. Based on the test data, we determine the optimal threshold which maximizes the F1-Score for the test data. For comparison, each model then was executed on the validation set, to asses the final performance. We can see that the Multilayer Perceptron with three hidden layers using MFCCs has the highest area under the curve of all models with 0.7741 which is the most general measurement for our purposes. The second highest value is achieved by the Multilayer Perceptron with three hidden layers using Mel spectrograms. Over all models, we see that none of the preprocessing techniques achieve constantly higher performances than the same model using different techniques. E.g. the largest area under the curve is achieved by using MFCCs with the Multilayer Perceptron, but MFCCs have the least area when it comes to LSTMs. Since the more complex models like LSTMs or CNNs are outperformed by MLPs, we can assume there is still improvement possible, since the more complex models can express the same or even a larger number of functions.

6.1 Statistical significance testing

In order to see if the Multilayer Perceptron with three hidden layers using MFCCs performs significantly better than the with Mel spectrograms, we perform a paired student's t-Test at 5% significance level. We use the paired test, since we are examining two procedures on the same data sample. As the null hypothesis, we assume that there is no statistically significant difference between the mean accuracies of the Multilayer Perceptron with 3 hidden layers using MFCCs and the one using Mel spectrograms. For the validation data, we make a prediction with each model and compute the accuracy for each sample, i.e. each chunk of music. A list of accuracies for each sample is then used to compute the p-value. As a result, the p-value is below the predefined significance level of 5% and thus we reject the null hypothesis. Hence, our Multilayer Perceptron with 3 hidden layers using MFCCs has a significantly higher accuracy than the Multilayer Perceptron with 3 hidden layers using Mel spectrograms.

6.2 Generating sheet music

To visualize the capability of the final model, we used a new MIDI file of Chopin's prelude op. 28 no. 7 and generated sheet music using MuseScore [6] shown in figure 12. Then, we synthesized a wave file from it and after feature extraction used the MFCC Multilayer Perceptron to predict the notes. The result shown in figure 13 was converted back to a MIDI file and processed by MuseScore again. Even as a non-musician it should be easy to see that the notations do not look really similar. At least one could notice a slight similarity in the range of keys used in each sheet.

7 FUTURE WORK

The realization of this projects highlights the possibility of exploiting "hidden" features for audio and digital music that can unleash tremendous potential in the field of Machine Learning. Though there are several possibilities for improvement. On the one hand, the pitch detection needs to be refined to achieve a proper transcription, on the other hand, properties like duration or volume need to be considered to create a more detailed sheet music. The sample dataset may seem small relative to the number of musical melodies trained, but once each file has been split in small segments of a second, the amount of data explodes and complicates the audio analysis. With more computational resources and time the following approaches can be considered to take and implement:

(1) Synthesize MIDI to Wave with different Sound Fonts that provide different sound wave patterns, for a 'more general understanding' of the sound.

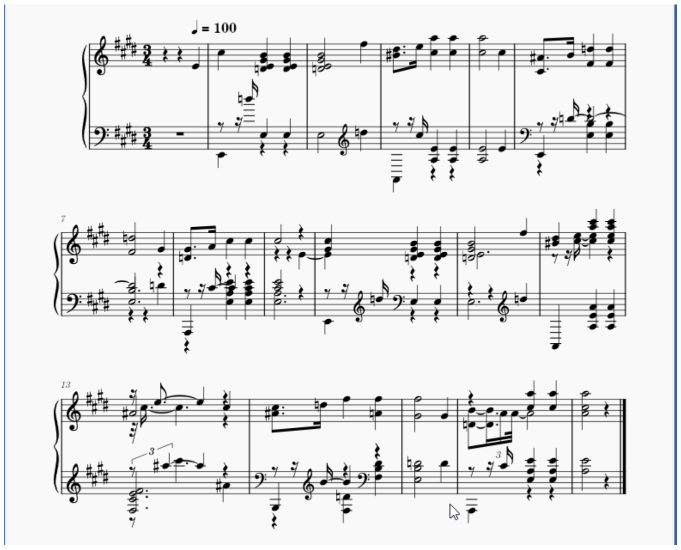


Figure 12: Sheet music based on original midi file

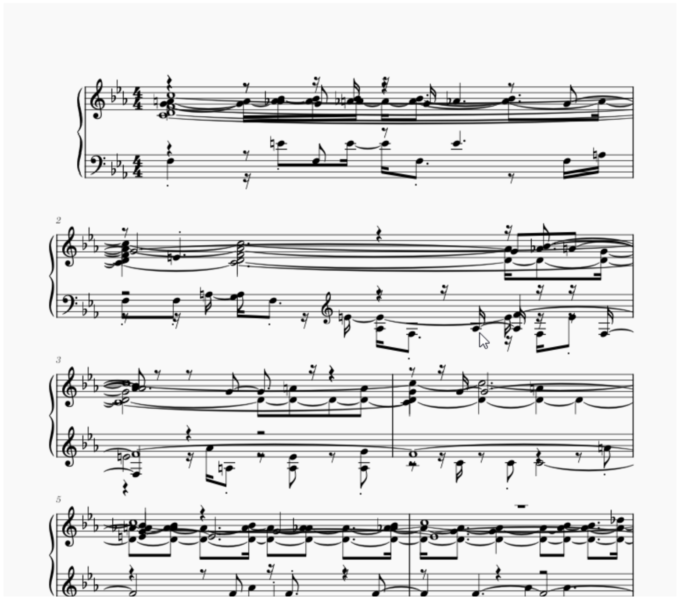


Figure 13: Sheet music predicted by the model

(2) Try retraining of more (state of the art) Neural Network architectures.
(3) Hyperparameter tuning of the input data: increasing or decreasing the size of the chunks, the number of frequency bands, the window size of the Fourier transform could yield a better performance
(4) Combining methods, e.g. for CNN's, there is a possibility to create new architecture and analysis by merging the three spectrums (MFCC, Mel, CQT) into a 9-dimensional feature vector. Each "image" would have three RGB vectors and when trained against an improved architecture, it is expected

that performance and model would drastically improve with enough training.

(5) Refine the models to not only detect the pitch of the notes played, but other properties like its tempo or duration, and even classify its genre.
(6) Use more MIDI samples and train a more general classifier with multiple instruments

8 CONCLUSIONS

We saw that music transcription can be tackled using deep learning techniques, although our models need some refinements to be usable as a sheet music generator. Since we created a pipeline to perform preprocessing including feature extraction and model training automatically, it will be easy to add new data and train other models. Furthermore, we could not see a major difference between the different feature extraction techniques on the same model architecture (note that the MFCCs were used to train the best model though). Hence, a combination of all feature extraction methods would be an approach interesting to explore in the future.

REFERENCES

[1] Md Zahangir Alom, Tarek M Taha, Christopher Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C Van Esesn, Abdul A S Awwal, and Vijayan K Asari. 2018. The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164* (2018).

[2] Ricardo F Alvear-Sandoval, José L Sancho-Gómez, and Aníbal R Figueiras-Vidal. 2019. On improving CNNs performance: The case of MNIST. *Information Fusion* 52 (2019), 106–109.

[3] Sercan Ö Arık, Mike Chrzanowski, Adam Coates, Gregory Diamos, Andrew Gibiansky, Yongguo Kang, Xian Li, John Miller, Andrew Ng, Jonathan Raiman, et al. 2017. Deep voice: Real-time neural text-to-speech. In *International Conference on Machine Learning*. PMLR, 195–204.

[4] Sameer Bansal, Herman Kamper, Adam Lopez, and Sharon Goldwater. 2017. Towards speech-to-text translation without speech recognition. *arXiv preprint arXiv:1702.03856* (2017).

[5] E. Benetos, S. Dixon, Z. Duan, and S. Ewert. 2019. Automatic Music Transcription: An Overview. *IEEE Signal Processing Magazine* 36, 1 (2019), 20–30. https://doi.org/10.1109/MSP.2018.2869928

[6] MuseScore BVBA. 2021. MuseScore 3.6. https://musescore.org/ Accessed: 2021-04-15.

[7] Jeremiah D Deng, Christian Simmermacher, and Stephen Cranefield. 2008. A study on feature analysis for musical instrument classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 38, 2 (2008), 429–438.

[8] Allen Huang and Raymond Wu. 2016. Deep learning for music. *arXiv preprint arXiv:1606.04930* (2016).

[9] Kyungdeuk Ko, Sangwook Park, and Hanseok Ko. 2018. Convolutional feature vectors and support vector machine for animal sound classification. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 376–379.

[10] Chang-Hsing Lee, Jau-Ling Shih, Kun-Ming Yu, and Hwai-San Lin. 2009. Automatic music genre classification based on modulation spectral analysis of spectral and cepstral features. *IEEE Transactions on Multimedia* 11, 4 (2009), 670–682.

[11] Gareth Loy. 1985. Musicians make a standard: the MIDI phenomenon. *Computer Music Journal* 9, 4 (1985), 8–26.

[12] Wei Ping, Kainan Peng, Andrew Gibiansky, Sercan O Arik, Ajay Kannan, Sharan Narang, Jonathan Raiman, and John Miller. 2017. Deep voice 3: Scaling text-to-speech with convolutional sequence learning. *arXiv preprint arXiv:1710.07654* (2017).

[13] Soumik Rakshit. 2019. Classical Music MIDI, Classical Music MIDI files for compositions of various composers. https://www.kaggle.com/soumikrakshit/classical-music-midi Accessed: 2021-03-15.

[14] Christian Schörkhuber and Anssi Klapuri. 2010. Constant-Q transform toolbox for music processing. In *7th Sound and Music Computing Conference, Barcelona, Spain*. 3–64.

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