

# Automatic Chord Detection

Daniel Gilkarov<sup>1</sup>

<sup>1</sup>School of Computer Science, Ariel University, Israel

dgilkarov33@gmail.com

## Abstract

Automatic Chord Recognition (ACR) aims to recognize musical chords played inside tracks, segmenting a track to different chords answering the questions when chords were played and what chords were played. ACR is a field with various uses among the music industry, with rapid growth in digital networks centered around music rises a great need for music analysis tools. One way of analysing music is through the chords that make it up, most western pop songs are composed with 3-4 chords. The necessity of ACR is assured through the inclusion of ACR as MIREX tasks since 2019. This paper presents a CNN based system for classifying chords in isolated recordings built upon previous work with robustness of the model to different musical instruments achieved using various data augmentation techniques. The model's ability are further demonstrated using two Out of Distribution (OOD) datasets with the first being a bit similar to the training data as it contains chords played by instruments already familiar to the model and the second dataset that's included as a testing dataset proposed by the article. It contains chord samples from "alien" instruments.

## 1. Introduction

In more modern times, with the internet - people use it among all its other uses for learning to play instruments, sharing their passion for music with others, sharing their music that they created with others [Jones \(2000\)](#). With a large amount of people that rely on the computer for learning their instrument, arises a need for an effective way to teach music, store music, play it using the computer.

In order to allow computers to store data about music, an intermediary called Musical Instrument Digital Interface (MIDI) was born ([Wikipedia contributors, 2023d](#)), a digital interface and "language" for representing structured music inside the computers memory.

Seeing as music is so popular, many companies and organizations were founded centered around it, with the music industry valued around \$26.2 billion [McCain \(2023\)](#) Radio stations, Music studios, Companies that develop applications that deal with music, such as instrument teaching apps [you \(2023\)](#); [sim \(2023\)](#), song recognizing apps [sha \(2023\)](#); [sou \(2023\)](#), audio streaming platforms [spo \(2023\)](#); [app \(2023\)](#) and more.

These are all organizations that have a need for a systematic way of characterizing music pieces ([Wikipedia contributors, 2023e](#)), for example, radio stations may want to choose which songs are most likely to be liked by their audience, music teaching apps usually work nowadays by listening to someone play and giving them feedback [you \(2023\)](#); [sim \(2023\)](#), and so these kinds of apps need a way to tell someone if they're playing correctly or not and if not describe what's the problem. One way of analysing a music piece is through the chords that make it up using **Chord Recognition**.

Most modern western music is composed using chords and melodies played with chords ([Wikipedia contributors, 2023f](#)),

knowing the chords of a song can help us in other tasks of music analysis, such as finding the key of a song, which is the group of notes, or scale that forms the basis of a music piece [Wikipedia contributors \(2023c\)](#). Knowing the key can help us group songs together, help us have a general understanding of how they'll sound, since the key dictates the notes that are used in the song, some songs change keys mid-song - some research is focused around detecting these changes in key [Chai & Vercoe \(2005\)](#). This kind of composition style is very popular with jazz music [Ulrich \(1977\)](#), with the feel of the song mutating and evolving as the song goes on.

A chord is a set of three or more notes that are played at the same time. Chords are built upon a certain note, known as the root note, which defines the chord, for example a D-minor chord has the note D as its root ([Wikipedia contributors, 2023a](#)). There are 2 major types of chords, major and minor, with major chords usually sounding happy and uplifting, minor chords which sound sad and melancholic ([Wikipedia contributors, 2023f](#)). The difference between them lies in the interval between the notes that make up the chord. An octave is the interval between one note and another with double its frequency ([Wikipedia contributors, 2023g](#)). Most musical scales are written so that they begin and end on notes that are an octave apart. For example, the C major scale is typically written C D E F G A B C. the initial and final C's being an octave apart. Tuning is the process of adjusting the pitch of one or many tones from musical instruments to establish typical intervals between these tones. The most common tuning system since the 18th century has been 12-tone equal temperament ([Wikipedia contributors, 2023b](#)).

Chord Recognition deals with recognizing chords played inside audio tracks, and providing information about which chords were played and when. It is a task which is sometimes given to people to do, stated as one of the Music Information Retrieval Evaluation eXchange (MIREX) tasks [mir \(2021\)](#). ACR on the other hand, tries to automate this process with the use of computers, with solutions proposed as early as 1999 [Fujishima \(1999\)](#).

This technique faces several challenges: songs usually have multiple instruments playing slightly different melodies, audio files are processed more than before, some chords and notes bear great similarity and machines some times struggle to discern them, to recognize the chord there's a need for identifying the root note and sometimes the notes are inseparable.

ACR has applications in various situations: It's used in song analysis, where the chords are extracted to help characterize it, because they sometimes play a big role in creating the whole experience of the song, it can be used to measure similarity between songs. Using it this way can assist in creating for example a song recommendation system, to recommend songs to users of an app that serves music based on their melody, portrayed through the chords. These kinds of techniques are also useful on their own in systems that aim to teach users to play an

instrument interactively, giving feedback on whether the right chord was played on various levels of verbosity. Audio search engines can use ACR to identify songs based on the chords that are recorded.

Of course this method is not without its downsides, many times a song isn't exclusively made up of chords, often there are melodies comprised of individual notes. This means that this method isn't enough on its own to paint a full picture, rather it is one tool in the toolbox of music analysis. Moreover, ACR usually relies on some kind of structure, both in that chords themselves are well-defined repeating entities, and also that the song, or music piece is composed with common sense (this can be argued to be inevitable), for example that there aren't 5 different chords overlapping at one point in the track, this makes it more difficult to deal with for instance, orchestral music, where not only it's impossible to say that from 03:05 to 03:10 a BMaj chord was played, maybe it's more like 03:05-03:07 - Bmaj, 03:02-03:09 - Amin, and so on.

ACR systems follow two distinct steps from start (Raw Audio input) to finish (Segmented timeline into chords labels and duration) most of the time: feature extraction and pattern matching. For feature extraction the audio signal is cut into short frames and each frame is transformed into an harmonic feature vector. Afterwards in pattern matching each frame is fit a chord label by measuring the similarity between its harmonic features and predefined chord models. Many systems also go further to introduce additional filtering steps aimed at dealing with errors that happen in the first and second steps, there's pre-filtering that takes place before the second step and post-filtering that happens after the second step. ACR solutions proposed in academic literature largely follow this structure, varying between different feature extraction techniques, different pattern matching techniques, with/without pre/post-filtering with varying techniques and different chord models ranging from the most basic model and some with great complexity.

Ever since ACR solutions started appearing, in terms of the feature extraction phase - chroma features are the most used features for it to this day. A chroma feature vector, also called pitch class profile (PCP), represents the energy distribution of a signal's frequency content across the 12 pitch classes of the equal-tempered scale. A temporal sequence of these chroma vectors is often called a chromagram. Another closely related option for representing audio with respect to time like PCP is a spectrogram, a spectrogram is a visual representation of the spectrum of frequencies of the audio signal as it varies with time. In practice these spectrograms are 2D representations of audio and are often used together with CNNs (Convolutional Neural Network) for ACR [Jadhav et al. \(2022\)](#), these CNNs have been found to deal with image data very effectively through the action of convolution, using kernels or filters that slide along input features and provide feature maps. As a chord is made up of a set of notes and it's assumed that the chord label can be determined by the occurrence of notes no matter in what octave, chroma features are still used to this day, and feature extraction is refined and improved through variation on the commonly-used chroma vectors. In terms of models used for the chord classification, there's an ever-growing use of deep-learning models such as CNNs, DNNs (Deep Neural Network).

For pre-filtering, most times the chromagrams are augmented before the chord labelling process, usually aiming to blur out noisy frames by smoothing the features across neighboring frames, typically done using moving average filter or moving median filter.

With regards to the pattern matching and post-filtering

phase, Some use discriminative ML (machine learning) techniques such as CRF (Conditional Random Field) [Burgoyne et al. \(2007\)](#), in recent years with the advancement in deep-learning there's more and more use of CRF along with some kind of deep-learning model for feature extraction: [Wu & Li \(2019\)](#) used a CNN for feature extraction along with CRF and BLSTM (Bidirectional Long Short-Term Memory), [Nakayama & Arai \(2018\)](#) experimented with a DNN for chord labelling and compared performance of various RNNs (Recurrent Neural Network) (LSTM, Bi-LSTM, GRU (Gated Recurrent Unit)) for post-filtering. Some use SVMs (Scale Vector Machine), in [Zhou & Lerch \(2015\)](#) SVM was compared against HMM (Hidden Markov Model) for post-filtering. there is repeated use of HMMs for post-filtering these describe the dynamic behavior of features in terms of transitions between chords. The transitions are dictated by a group of probabilities called transition probabilities that describe the likelihoods of transiting from a chord to another. An algorithm called Viterbi finds the most likely chord sequence by evaluating the joint probability of the output from the pattern matching stage and all possible chord sequences based on a given transition matrix. This in turn reduces the number of false chord predictions by restricting unlikely chord transitions.

### 1.1. Our Contribution

- **Robust Model:** This article offers a robust approach for ACR capable of recognizing chords played in various different instruments in contrast to the baseline model which is only trained on guitar recordings and shown to very ineffective with instruments that aren't guitar.
- **Quicker Training Time:** The proposed approach is shown to be able to train very quickly in comparison to the baseline (5 epochs vs. 15 epochs) - at the same time having performance on par with the baseline in the data it is specially trained for (guitar recordings).
- **Testing dataset with various instruments:** The article offers a dataset of 80 recordings played on 8 different instruments which usually are not available - for example: Clarinet, Harp, Flute - this dataset provides a great way to prove robustness of a chord classification model due to the various rare instruments present in it.

### 1.2. Paper Structure

The remainder of this paper is structured as follows: Section 2 surveys related work mainly regarding common ACR approaches in general, and Deep Learning (DL) solutions in particular; Section 3 discusses datasets used in this paper and the pre-processing procedure; Section 4 presents the different data augmentation methods that were utilized in this work; Section 5 presents the approach employed in this paper; Section 6 presents a fine-grained experimental evaluation process, that consists of an evaluation of our augmentation-based ACR approach, and the results of the comparison between our approach and a CNN baseline version; Finally, Section 7 concludes the paper and offer some directions for future work. For ease of reading, Table 1 provides a list of abbreviations that are commonly used in this paper.

## 2. Related Work

Initially, Chord Recognition was performed by calculating chroma features ([Fujishima, 1999](#)), then labeling chords by

Table 1: *List of abbreviations*

Abbreviation	# Meaning
ACR	Automatic Chord Recognition
BLSTM	Bidirectional Long Short-Term Memory
CE	Cross Entropy
CNN	Convolutional Neural Network
CRF	Conditional Random Field
DL	Deep Learning
DNN	Deep Neural Network
GRU	Gated Recurrent Unit
HMM	Hidden Markov Model
MIDI	Musical Instrument Digital Interface
MIREX	Music Information Retrieval Evaluation eXchange
ML	Machine Learning
PCP	Pitch Class Profile
RMS	Root Mean Square
RNN	Recurrent Neural Network
SNR	Signal to Noise Ratio
STFT	Short-time Fourier transform
SVM	Scale Vector Machine

template matching against models of known chords. For the most part, features used in the field are chroma vectors and Tonnetz Euler (1739) representation. For pattern matching and chord sequence decoding, Gaussian mixture models with feature smoothing Cho (2014); Cho & Bello (2011) and HMMs Sheh & Ellis (2003); Ueda *et al.* (2010) have been the most popular choices, respectively.

Chroma features, despite being very widely used in this field tend to be noisy, and in the context of specifically looking for chords, sometimes these chroma features contain scattered notes which can't be recognized as chords. This leads other researchers to incorporate different types of auxiliary features, such as temporal continuity Weller & Ellis (2003); Mauch & Dixon (2010). Other features used include various musical properties such as beats Mauch & Dixon (2010); Papadopoulos & Peeters (2011), key Pauwels & Martens (2014), bass Sumi *et al.* (2008).

In Cho & Bello (2014) the importance of different features and data manipulation techniques for ACR are systematically investigated, revealing a great importance for using pre and post filtering, where in pre-filtering typically blurring out noisy frames by smoothing the features across neighboring frames before the pattern matching phase. post-filtering is usually applicable in HMM-based chord recognition systems. The above methods are shown to greatly improve accuracy.

A variety of chroma features have been proposed for ACR and other tasks. Typically a pitch feature is first derived from which a basic chroma feature is calculated by summing elements of the same pitch class. The pitch feature is usually based on a Constant Q-transform (CQT), a spectrogram with a log-frequency scale with an equal number of frequency bins per octave. Alternative features to this basic chroma are easily derived through transformation of the pitch feature. Well-known transforms for chroma include log-compression which de-emphasises large energy peaks and spectral weighting which places more weight on the central elements of the pitch scale, thereby de-emphasising the effect of higher overtones.

The use of DL models is seen to increase in recent years and show promising results in ACR, usually used as part of a pipeline for feature extraction - Humphrey *et al.* (2012) employed a CNN to extract Tonnetz features from audio data. Korzeniowski & Widmer (2016a) adopted a DNN to compute chroma features. CNNs are very often used Humphrey & Bello

(2012); Korzeniowski & Widmer (2016b) - showing performance not far from the state-of-the-art. CNNs are often used along with HMMs Humphrey & Bello (2015); Zhou & Lerch (2015). CNNs often come with a computational expense that's associated with CNN's, O'Hanlon & Sandler (2021) aims to tackle this issue by suggesting what they call a "FifthNet", in their words a compact CNN that achieves similar results to other CNN-incorporating systems with much lower computational requirement.

Another popular network architecture used a lot for ACR is RNN. Boulanger-Lewandowski *et al.* (2013); Sigtia *et al.* (2015) explored a RNN as chord sequence decoding method, relying on deep belief network and a DNN, respectively. RNNs are also used with a language model which predicts only the sequence of chords without considering their durations. This might be helpful when there's a large vocabulary. A large-scale study of language models for chord prediction was conducted in Korzeniowski *et al.* (2018). Without audio data, the authors trained just a language model with the chord progression data only and showed that RNNs outperformed N-gram models. A later publication Korzeniowski & Widmer (2018) combined the RNN-based harmonic language model with a chord duration model to complete the chord recognition task. Wu & Li (2019) trained a CNN feature extractor with large MIDI data and combined BLSTM with CRF as a sequence decoder. In recent years there's an ever-expanding increase in use of transformer networks. Use of Transformers has become very popular, achieving the state-of-the-art results in various domains. A well-known example is bi-directional encoder representations from Transformers (BERT) Devlin *et al.* (2019). BERT is a pre-training model based on masked language model for language representations that achieved state-of-the-art results on eleven NLP tasks. Transformers were shown to succeed in the music domain as well, Huang *et al.* (2018) proposed music Transformer for symbolic music generation. Music Transformer employed relative attention to capture long-term structure effectively, which resulted in music compositions that are both qualitatively and quantitatively better structured than existing music generation models. Park *et al.* (2019); Chen & Su (2019) proposed Transformer models to perform ACR with good results, Such sequence based networks are often considered to process a music language element.

In this article we take a chord classification system, designed for classifying short recordings of guitar sound to a range of 10 chords Shah (2020). In these recordings there isn't an aspect of segmenting the audio to different chords as in some ACR systems. A model is trained for giving labels out of 10 options to an audio signal, this kind of system learns knowledge which can be leveraged for traditional ACR as seen in Nadar *et al.* (2019). This chord recognition system has good results classifying guitar chords from a dataset curated by MONTEFIORE RESEARCH GROUP of university of Liège - recall: 91%, precision: 94%, f1-score: 92% We aim to make this system more robust to different sound other than guitar, for example: the baseline model performed poorly on piano recordings from the same dataset - recall: 21%, precision: 22%, f1-score: 21%

### 3. Datasets

The datasets used in this paper are summarized in Table 2, it provides the total dataset duration and total number of chord segments.

Table 3 provides amounts of samples of each class for each dataset

In this article, 3 datasets were used, one in training and two for testing:

1. **Osmalsky**: a dataset created by MONTEFIORE RESEARCH GROUP of university of Liège [Osmalsky et al. \(May 2012\)](#); [Nadar et al. \(2019\)](#); [Shah \(2020\)](#) is used for training the model. The dataset is a free publicly available collection of isolated recorded samples of chords played on guitar, accordion, violin and piano. This dataset is comprised of 2 second long WAV recordings sampled at 44100 Hz.

Originally the creators of the datasets meant this dataset to be a guitar chords dataset, with the rest of the recordings of the other instruments meant to be used as a test set to prove their ML algorithm's ability to generalize to other instruments, in this article however we will make use of these samples to train our DL model - to counter the fact that the dataset is greatly unbalanced between the guitar recordings and the others, we utilize data augmentation. The chord recordings are distributed equally between 10 different chord classes:

A, Am, Bm, C, D, Dm, E, Em, F, G

There are 200 chord recordings of each class for the guitar recordings and 10 recordings of each class for the other instruments. This dataset (like a lot of others with the same niche) was created with the aim of matching to some degree the sound of western pop music, in their article they claim this was also the motive behind choosing these chords specifically and the guitar as the main recording instrument.

The guitar recordings were recorded using 3 different acoustic guitar(s) and a nylon string guitar - half in an an-echoic chamber and half in a noisy environment in order to simulate real life scenarios for artists recording (Studio and Live performances), moreover the recordings were played in several different playing styles - (arpeggio, staccato, legato, etc.). These varieties in recordings are meant to offer a more wide range of sound to help models learning this data become more robust.

2. **IDMT-SMT-Chords** : Synthetic dataset [Nadar et al. \(2019\)](#) created using Ableton Live and Garage Band, the dataset consists of 2 second long samples of MIDI chords. The chords are played in the following chord voicings which are commonly used on keyboard instruments and guitars:

Ableton Live:

- Campfire Guitar
- Guitar Open
- Nylon Concerto Guitar
- Celestial Pad
- Grand Piano Reverb
- Grand Piano
- Piano to Pad
- Sadness Pad
- Sweetness Pad

Garage Band:

- Acoustic Guitar
- Hard Rock Guitar
- Classic clean Guitar
- Boesendorfer Grand Piano
- Deluxe Modern Piano

- Jazz Organ
- Steinway Grand Piano

The dataset was created to remedy the scarcity of datasets with 7th chords but this article doesn't aim to classify 7th chords and therefore only the chords which are relevant here were extracted from the recordings. Out of the **7398** audio segments we end up using **160**, as mentioned in table 3 we have 6 audio samples of every class generated using 6 different *guitar* sounds available in the programs mentioned and 10 audio samples of every class generated using 10 different *piano/keyboard* sounds, one sample per each different chord voice.

While the samples in the training set are very organic (being played by a human on a real instrument and recorded in simulated scenarios) and vary mostly through the way the chords are played (arpeggiated, strummed, etc.), its is apparent that in contrast the test set is created artificially (which standardizes the sound produced) contains samples with great in-dataset variety due to the fact that for every chord class we have one sample of each instrument preset in the MIDI software with some presets producing sounds resembling synthesizers. This means the test dataset is very different from the training data - this is good since the testset being very different from the training set provides a better test for robustness of a ML model.

3. **Multistrument** : This dataset is a diverse dataset that was created for the purpose of further validating the robustness of the proposed models. The dataset was created using the FL studio software to create chords sounds from samples of various instruments available in [VSC \(2023\)](#) - a freely available sample library recorded by volunteers. During creation - 8 instruments were picked out of the available options with diversity and variance in mind - wind instruments, brass instruments, etc. were picked due to the previous datasets and datasets available online lacking representation with those kinds of instruments - in fact most chord classification datasets offer guitar and piano recordings for the most part, this is probably not very shocking as these are 2 of the most popular instruments that are used as well for chord music. The instruments used in the dataset are presented now:

- Clarinet
- Flute
- Harp
- Horn
- Oboe
- Trumpet
- Tubular Bells
- Xylophone

From every instrument all 10 chords were generated. This dataset further validates the robustness of the models presented as the instruments present in it weren't at all part of the training process of the models and so if the models are shown to succeed in classifying these recordings it proves the models were able to not only get good results with their training data but also perform well on data that was unseen before during training.

### 3.1. Pre Processing

The preprocessing phase starts off by taking the raw WAV files, constructing a dataframe where each row represents one sample in the dataset. For each sample the label is also included, one of the classes detailed in section 3.



Dataset	Duration (h)	#Chord Samples	Sample Duration	Sampling Rate
Osmalsky	1.3	2400	2 seconds	44.1 KHz
IDMT-SMT-Chords	0.08	160	2 seconds	44.1 KHz
Multistrument	0.04	80	2 seconds	44.1 KHz

Table 2: Overview of the chord recognition datasets with the total duration in hours, the number of chord samples, sample duration per dataset and sampling rate. In order to enlarge the training dataset (Osmalsky), we use time-stretching (2 variants) and additive gaussian noise (3 variants) as data augmentation technique. Hence, each original file results in 5 augmented files including the original recording.

Dataset	Guitar	Piano	Accordion	Violin
Osmalsky	200	60	60	60
IDMT-SMT-Chords	6	10	0	0

Table 3: Overview of amounts of chord samples per instrument for each dataset

### 3.1.1. Mel Spectrogram

The main feature that’s fed into the CNN model propped in this article is a mel-spectrogram. Mel-spectrograms are spectrograms where the frequencies are converted to the mel scale. Humans hear frequencies logarithmically rather than linearly. And so, a 100Hz difference in the Mel Scale corresponds to what a human would typically perceive in the actual world. So the mel-spectrogram deals with the issue of the spectrogram not reflecting change in frequency as a human would, the mel-scale converts a tone’s perceived frequency to its actual frequency. Mel spectrograms hold sound information which the human ear could perceive. The Mel scale and Hertz(Hz) are related by the given formula:

$$m = 2595 \log_{10}(1 + \frac{f}{700})$$

For every sample a mel-spectrogram is calculated (window length - 2048) using the librosa python library which calculates a Short-time Fourier transform (STFT) . The largest shape of these spectrograms present in the whole dataset is found and the spectrograms are zero padded from the right to achieve a uniform shape. Every mel-spectrogram is scaled using the log function Choi et al. (2021) in the following way:

$$\log(X + \alpha)$$

where alpha is an arbitrary small number like  $10^{-7}$  or 1 and X is an input mel-spectrogram. This article uses  $\alpha = 10^{-7}$ .

### 3.1.2. Data Normalization

This article takes advantage of data normalization to allow the model to converge more quickly by reducing the variety of spectrograms frequencies across different instruments. Normalization makes the data distribute in a more gaussian fashion, therefore making the data more balanced and prevents outliers which makes the data skewed and often hurts the learning process. The log-mel-spectrograms are normalized so that all the values lie in the  $[-1, 1]$  by taking each row of a spectrogram and dividing every value by the maximum value of the row:

$$X_{normalized} = \frac{X}{\max(X)}$$

## 4. Data Augmentation

Data augmentation Wei et al. (2020); Abayomi-Alli et al. (2022) is a common technique that is used to artificially increase the size of a dataset by creating modified versions of existing data. In the realm of audio, this can be done by applying transformations such as pitch shifting, time stretching, or adding noise to the audio clip. Data augmentation is often used in machine learning and especially deep learning where most of the time the crucial make-or-break point of a DL model is it’s access to good quality diverse data. Data augmentation is used to improve the performance and generalization of models by providing them with additional training examples. The new augmented data is viable since the augmentations do not interfere with the transcription of the original audio-recording and usually can be label-perserving which is important for supervised learning systems. The new data can be used to further train the model. Focusing on a minimal number of high-impacting augmentations which introduce variance that’s maybe lacking in the training data is vital to successfully improving the model, when using augmented data - being able to maintain good performance on test data which wasn’t augmented is vital since if there’s success with augmented data but failure with real data - there isn’t real value in such improvements. In this work, two augmentation methods are considered:

- Gaussian-Noise Taenzer et al. (2019) (section 4.1)
- Time-Shift Salamon & Bello (2017); Madhu & Kumaraswamy (2019) (section 4.2)

### 4.1. Gaussian Noise

Artificially adding Gaussian-Noise augmentation to the training data can improve “ the robustness and generalization of an ACR model. The goal of this method is to make the model more resistant to variations in the input data such as the differences between sounds of different instruments, in addition it helps by simulating real-world scenarios where the data is noisy in contrast with studio-recorded data with perfect conditions. In order to augment the audio-recording and add noise, first noise is generated based on a certain Signal to Noise Ratio (SNR) value - SNR is given by:

$$SNR = 10 \log(\frac{RMS_{signal}^2}{RMS_{noise}^2}) \quad (1)$$

where for a group of n values involving x1, x2, x3,... Xn, the Root Mean Square (RMS) is given by:

$$RMS = \sqrt{\frac{x_1^2 + \dots + x_n^2}{n}} \quad (2)$$

when generating noise, given some SNR value, the  $RMS_{noise}$  is calculated and random gaussian distributed values are gener-

ated with standard deviation equal to it, then adding the original signal and the noise arrays resulting in the original audio with noise added to it with the volume of noise dictated by the SNR value.

#### 4.2. Time Stretching

The audio sample is speeded up or slowed down without changing pitch. This prevents the model from overfitting to a certain tempo when regarding music in general and more specifically, chords played on string instruments, by creating audio samples which are faster (by a factor of 1.07) and slower (by a factor of 0.81) the aim is to increase general robustness - the model will have to be able to deal with the audio with varying speeds and therefore be more ready to tackle real-life audio samples with varying speed. The time stretching is implemented using `librosa.effects`

### 5. Framework

The following section describes the method proposed by the article for doing ACR on audio samples containing one chord - labeling them as one of predetermined classes, see section 5.1 for reference. In section 5.2 the general structure and data-flow of the models is laid out. In section 5.3 the models are discussed with greater detail, also presenting the motivation for the proposed actions.

#### 5.1. Classification Classes

The classification classes for the chords in this system are:

A, Am, Bm, C, D, Dm, E, Em, F, G

#### 5.2. System description

The main architecture of the model has three main building blocks (See figure 1 for reference):

- **Pre-Processing** The model receives raw audio file in WAV format as input and it calculates a mel-spectrogram of it, then applying log to every value in it and finally normalizing it to the range [-1, 1]. See section 3.1.
- **Data augmentation** Every sample (except samples of guitar sound) in the processed dataset is applied Time Stretch and Gaussian Noise data augmentations to get in turn 5 different augmented samples of each sample. See section 4
- **Chord Classification** The data is input into the CNN where it gets classified as one of the 10 predefined classes defined above.

#### 5.3. Model Architecture

The baseline model which this article builds upon [Shah \(2020\)](#) consists of a CNN. This article proposes two models and investigates their performance side by side. The first model is identical in its architecture to the baseline model, the second model is similar with the dropout [Srivastava et al. \(2014\)](#) layers taken out.

The motive behind removing dropout from the architecture is that dropout's purpose is to counter-act overfitting but it comes with a cost of having to train the model for a bigger amount of epochs - it's a tradeoff between train-time and overfitting amount, so removing dropout allows a model to train and converge much faster and to counter-act overfitting we want to use

something other than dropout and that's where data normalization comes in. The first model (with dropout layers) is detailed in figure 4 The second model (without dropout layers) is detailed in figure ??

Described through words - the model is a CNN, taking as input the spectrograms that were computed for each sample. The input first passes through a convolution [Yamashita et al. \(2018\)](#) layer with a relatively small filter of size 5x5 with a stride of 1 unit. This convolution is followed by a max pooling operation with a stride of 4,2 and afterwards the output is fed through ReLu [Agarap \(2019\)](#) activation. This process is repeated once more, with a convolution, max pooling and a ReLu activation after which another convolution happens followed by ReLu activation where the outputs are then flattened and fed through fully connected layers as is customary with these kinds of model architectures. At the end of the chain we have a softmax activation layer (as is very popular with multi-classification systems) that outputs a 1-dimensional probability vector that contains the probability of each class in the vector - thus classifying into our 10 classes that were mentioned before.

##### 5.3.1. Cross Entropy Loss

The choice of loss function for the models **CNN** and **CNN.nodropout** is Cross Entropy (CE) [Martinez & Stiefelhaugen \(2018\)](#) - for this purpose - the labels of the chords are one hot encoded before training. The CE loss compares one hot encoded vector  $y$  with a probability vector  $p$ , both of them in a logarithmic and exponential fashion. One hot encoded vectors are vectors of size #classes where all but one entry are zero and that single entry is 1, corresponding with one of the classes and this acts as a label. The CE function is found in equation 5 and is defined with  $M$  equal to #classes

$$ReLU(x) = \max(0, x) \quad (3)$$

$$softmax(x) = \frac{e^x}{\sum e^x} \quad (4)$$

$$CrossEntropy = - \sum_{i=1}^M y \log(p) \quad (5)$$

### 6. Experimental Evaluation and Results

The following section is dedicated to the validation of our hypothesis that using data augmentation and normalization can result in a robust model that's able to perform ACR on OOD data. The evaluation was conducted on an Ubuntu 64Bit OS with Intel(R) Xeon(R) Silver 4214R CPU, 128GB RAM and NVIDIA RTX A5000 GPU, using TensorFlow (v2.11.0) as backed for Keras (v2.11.0).

In all of the experiments, the following training parameters were used: Adam [Kingma & Ba \(2017\)](#) optimizer,  $1e-3$  learning rate, batch size of 20. The experiments compare between 3 different options for training the CNN - With Dropout and without normalization, with normalization and without dropout and finally, dropout and normalization together.

#### 6.1. Evaluation Metrics

In order to measure how good a model is, there are many different metrics that can indicate the quality of a model. Specifically the case of this article warrants ways to measure success with

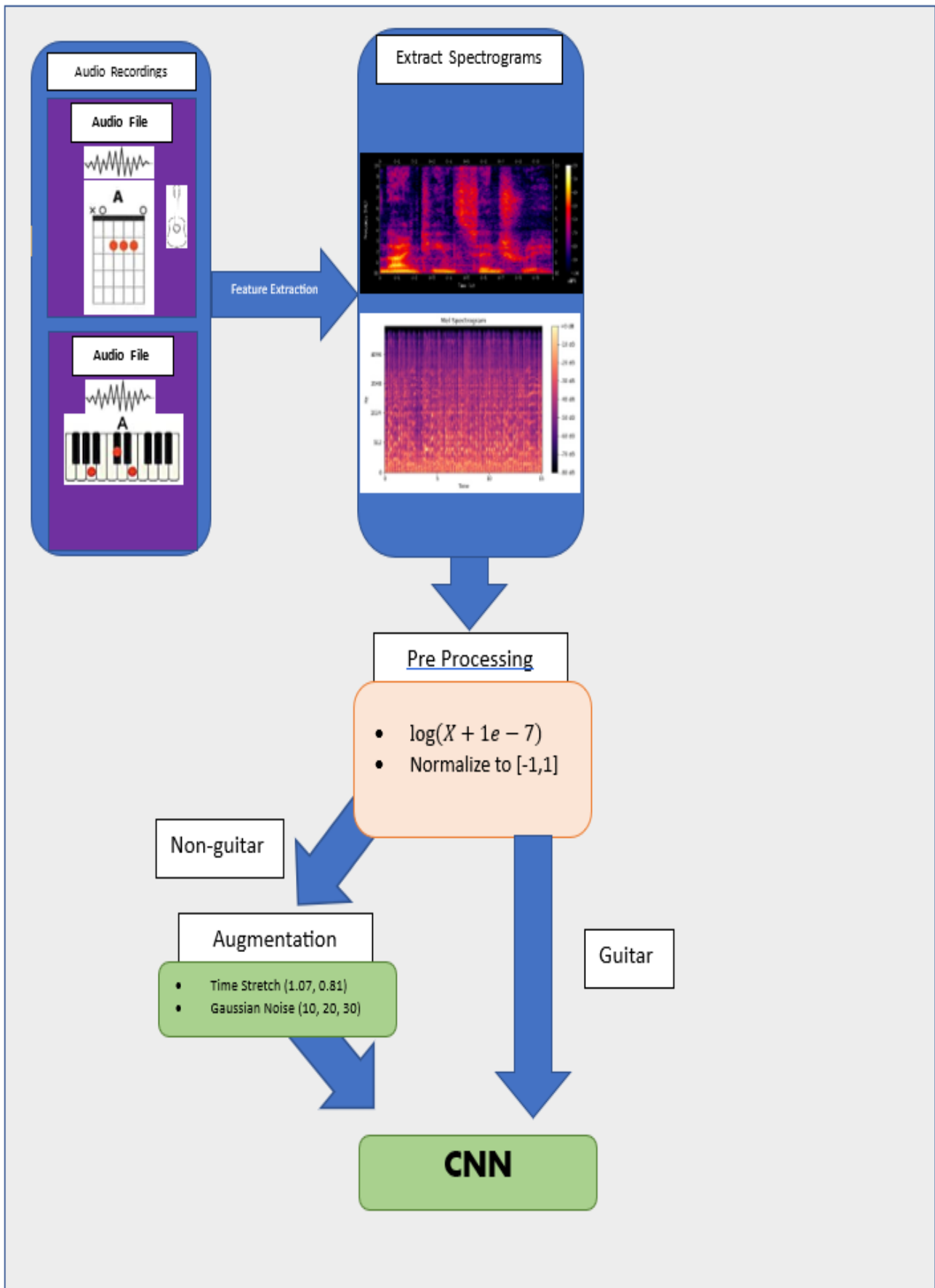


Figure 1: An illustration of the suggested framework in this paper. For any audio recording of musical instruments, the mel-spectrogram is extracted as detailed in section 3.1.1. The mel-spectrogram is transformed using the log function as detailed in section 3.1 and then normalized to the range [-1, 1]. The data is then augmented using Time Stretching and Gaussian Noise addition and finally fed into the CNN.

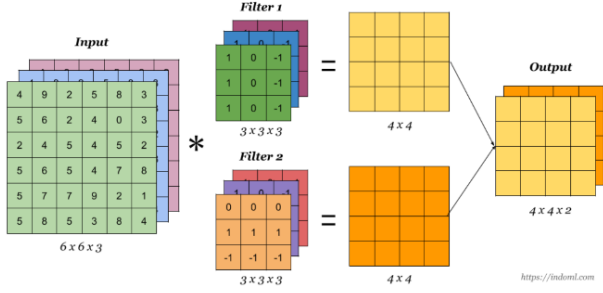


Figure 2: Convolution operation - the basic operation used in Convolutional Neural Networks.

classification tasks as this is what's done in this article. For classification problems with a balanced ratio of the classes present in the training dataset, accuracy is good enough and can indicate quite well how good a model is at a certain task, accuracy is given by equation 6

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (6)$$

where True Positive (TP) is the number of inputs which were true and the model also classified as true, True Negative (TN) is the number of inputs that were false and the model classified as false. False Positive (FP) is the number of inputs that are false but the model classified as true and False Negative (FN) is the number of misclassified negative samples.

Precision is a measure of how many of the positive predictions made are correct and it's formula is given by equation 7

$$Precision = \frac{TP}{TP + FP} \quad (7)$$

Recall is a measure of how many of the positive cases the classifier correctly predicted, over all the positive cases in the data its formula is presented in equation 8, as follows:

$$Recall = \frac{TP}{TP + FN} \quad (8)$$

In addition to these metrics there exist another, called F1-score, which is a better metric for classification performance measurements in imbalanced datasets [Jeni et al. \(2013\)](#). F1-score is regarded as harmonic mean of Precision and Recall, representing the two metrics together, F1-score can only be high when both precision and recall are high so in a sense it is a more guaranteeing metric than either one of them alone. F1-score is given by equation 9:

$$F1 - score = 2 * \frac{Recall * Precision}{Recall + Precision} \quad (9)$$

These metrics can still falter when determining success on data that varies greatly and since it projects a general picture without detail about different instruments in this case, cases where for example a model has very great success with all instruments excepts piano - let's say it classifies guitar, accordion, violin with absolute success and always fails on piano - we would then get accuracy percent of 95.8% - this presents a false sense of great success - in this case we need to look at metrics of the model on each instrument voicing separately.

## 6.2. Experiments

The following experiments are meant to prove that the data augmentation, normalization offered in this article has value and to validate the contributions this article claims it makes. The main results that are needed to be proved are - robustness which will be validated by the OOD datasets, and the value of **CNN\_nodropout** remains to be exposed through the experiments. In all experiments the first dataset 2 is split with 0.65 ratio into train/test. All experiments contain a table marking the performance of the model tested on the test split set. Below is a list ordering the experiments:

1. Experiment 1 - normalization, CNN - first of all assess the effects of normalization.
2. Experiment 2 - no normalization, CNN - assess the performance of CNN - with dropout to act as a baseline to prove model without normalized inputs doesn't converge as fast as the one with normalization.
3. Experiment 3 - normalization, CNN\_nodropout - Here we can expect the model without dropout will converge quicker than the one in experiment 1, and this is presented as one of benefits offered by the article.

The collective data gathered from the experiments with all the main results is located in table 4

### 6.2.1. Experiment 1

First experiment - CNN - normalization

This experiment serves as a baseline, to first of all assess that the normalization is indeed producing results as expected - faster convergence - this is portrayed through the fact that without normalization [Shah \(2020\)](#) trained the model with the dropout layers for 70 epochs until it converged to a desirable state.

Instrument	Accuracy (%)	Recall (%)	Precision (%)	F1-Score (%)
Guitar	<b>94</b>	94	95	94
Accordion	100	100	100	100
Violin	100	100	100	100
Piano	<b>98</b>	96	100	98

We see from the table that the model has great success with accordion, violin, piano and lesser success with the guitar samples - this is to be expected for 2 reasons - no augmentation was done to the guitar samples and so the model becomes better at the other instruments, there are way more guitar samples and so when classifying a larger volume of samples there is a lesser chance of having perfect scores as opposed to - let's say getting perfect score on 10 samples.

### 6.2.2. Experiment 2

Second experiment - CNN - **no** normalization

This experiment is supposed to highlight the importance or effect of the normalization on the ability of the model to converge within 15 epochs (for this matter)

Instrument	Accuracy (%)	Recall (%)	Precision (%)	F1-Score (%)
Guitar	52	34	86	48
Accordion	96	89	98	93
Violin	98	95	100	97
Piano	60	33	76	46

As evident by the table, without the normalized data the model just can't learn within 15 epochs whereas the model that



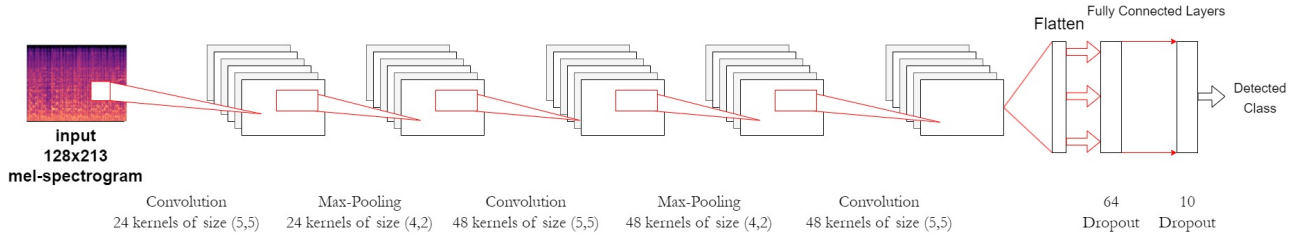


Figure 3: Proposed architecture with dropout layers

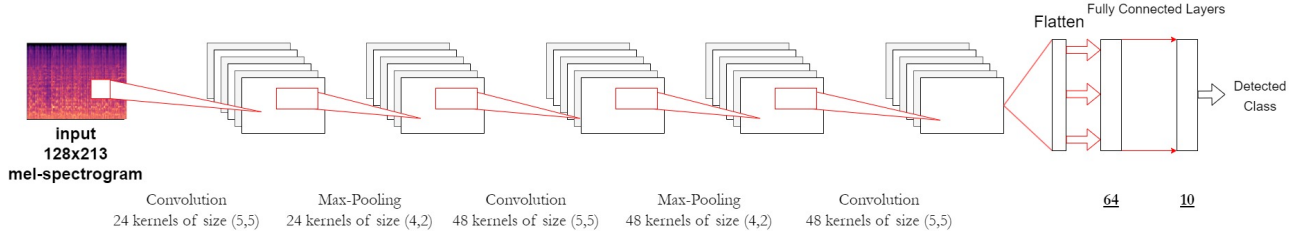


Figure 4: Proposed architecture without dropout layers

learned normalized data could even have started overfitting and could be stopped even earlier - the equal epoch amount is meant to make the experiment as scientific as possible - singling out only one difference between the 2 experiments and deducing that the changes we see between the experiments stem from the single difference that was set between them. What's interesting to note is that the model did succeed in learning the violin/accordion samples.

### 6.2.3. Experiment 3

Third experiment - CNN<sub>nodropout</sub> - normalization

This experiment shows the potential of the model without dropout layers - but here there's a need to exercise caution - the normalized data helps to prevent overfitting but too many epochs and the model overfits, this means there's a need here to utilize early stopping (normally would be done using a validation set - stopping the training once the loss in the validation set starts increasing), this won't be the case, instead the model will learn for 6 epochs, the true question and the true proof lies within testing these models on the OOD datasets.

Instrument	Accuracy (%)	Recall (%)	Precision (%)	F1-Score (%)
Guitar	90	90	92	91
Accordion	100	100	100	100
Violin	100	100	100	100
Piano	98	96	100	98

Looking at the metrics here, the model reaches satisfactory performance - a bit behind the CNN model but it trains in less than half the time. Of course there's a chance the training can continue without overfitting but for now, until the models are tested on the OOD data, these experiments stand as a proof of concept regarding the data normalization's efficiency, the dropout model gets a "skeptical" treatment and the real proof for it will stand with the OOD trials.

### 6.3. OOD performance

In order to better assess our models integrity and robustness we can do some thing called OOD (Out Of Distribution),

Table 4: Table of general performance on the test set during experiments

Experiment	No. Epochs	Accuracy (%)	Recall (%)	Precision (%)	F1-Score (%)
(1)	15	95	95	96	95
(2)	15	58	42	88	56
(3)	6	92	91	93	92

basically - ML algorithms and DL algorithms have a tendency to overfit the data they're learning, where they become great at classifying the data they trained on but they learn to be good at that specific data and they don't learn a general rule about the data, rather something true in this specific case. This is one of the biggest challenges of ML and DL - various techniques have been found to reduce overfitting such as regularization, dropout - to name a few. When we look at a model we say to ourselves - well as we can see the model was very successful at the data it learned, but the real question and the test is did the model learn something general that can be applied for different data from different distributions other than the one it knows very well from learning. OOD regards to analyzing a model's behaviour when fed with data out of its known and trained upon distribution. Part of the trials that take place in this article included taking OOD datasets and seeing how the proposed models fared with them - the models were checked with 2 OOD datasets: a dataset of midi generated musical recordings of a similar structure to the one we learned [Nadar et al. \(2019\)](#) - this data has some major differences and that's beneficial in order to truly test the models - take it to an extreme. The data is computer generated whereas the training data was played and recorded with real instruments and real environments, moreover the dataset contains various tones of piano recordings and guitar recordings, there are 100 piano recordings ranging from different tones like - "grand piano", "sadness pad", "jazz organ". There are also 60 guitar recordings ranging from different tones like - "acoustic guitar", "hard rock", "campfire guitar". A second dataset was created specially for this article - tailored to really push the models to the edge in terms of their ability to generalize by choosing instruments with a very different nature than the ones used in training - wind

instruments (Clarinet, Oboe) as opposed to string instruments like guitar and violin. The dataset contains one chord of each class for each instrument, this is to ensure even coverage of all classes with a wide spectrum of sound recorded using real instruments to add an element of real-world audio. For some reference the tests included running a CNN architecture exactly like the original one - but trained with the augmented and normalized data. Below are the insights:

The comparison here is between Model 1 (CNN) and Model 3 (CNN\_nodropout) from table 4. There is no reason to look now at models with unnormalized data as it was already shown that the models outperform the model which learnt unnormalized data. Going back to Model 3 - now that we test the model using the first OOD dataset we can tinker the amount of epochs and find a sweet spot - where it does well on the training set but it also generalises to the OOD set, after checking a few values - 5 epochs seem to yield good results:

Table 5: OOD performance on dataset 3 for Model 1

Instrument	Accuracy (%)	Recall (%)	Precision (%)	F1-Score (%)
Guitar	78	79	79	79
Piano	75	76	74	75

Looking at the results model 1 has - the one with dropout layers - it is faring not so bad considering this OOD dataset is quite different from the training set.

Table 6: OOD performance on dataset 3 for Model 3

Instrument	Accuracy (%)	Recall (%)	Precision (%)	F1-Score (%)
Guitar	<b>88</b>	90	88	89
Piano	81	86	84	85

Seeing as the model without the dropout layers did well, and not only that - it did better than the model with dropout layers, and while training 10 epochs less, this means this model proved to be more robust, and learn more quickly. Moving on the models are tested on dataset 3, see table 8, it is apparent that the first model does bear success with instruments which it didn't have learning experience with - this already indicates the model to be robust across wind instruments and string instruments, however when presented with audio generated by bell type instruments it fails.

Table 7: OOD performance on dataset 3 for Model 1

Instrument	Accuracy (%)	Recall (%)	Precision (%)	F1-Score (%)
Clarinet	<b>90</b>	90	85	86
Flute	<b>90</b>	90	85	86
Harp	80	80	75	76
Horn	<b>90</b>	90	85	86
Oboe	<b>90</b>	90	85	86
Trumpet	<b>100</b>	<b>100</b>	<b>100</b>	<b>100</b>
Tubular Bells	50	50	45	46
Xylophone	20	20	7	10

## 7. Conclusions and Future Work

The goal of this paper was to improve an ACR model, taking it from being a model that only has success classifying guitar samples to a model that has success classifying guitar better than the original and moreover, has success classifying piano, violin, accordion. The newly-attained robustness that wasn't

Table 8: OOD performance on dataset 3 for Model 3

Instrument	Accuracy (%)	Recall (%)	Precision (%)	F1-Score (%)
Clarinet	80	80	70	73
Flute	<b>90</b>	90	85	86
Harp	90	90	85	86
Horn	<b>90</b>	90	85	86
Oboe	<b>90</b>	90	85	86
Trumpet	90	90	85	86
Tubular Bells	50	50	33	38
Xylophone	30	30	15	20

present before was further verified using 2 OOD datasets that are different than the training data - in the way they were created, in the variance between single samples, etc. Two models were presented with one outperforming in some places and of course the original in success on the OOD datasets and training time (Original model was trained for 70 epochs while the proposed model was trained for 5 - this means a 14 fold decrease in training time). Part of the success is due to data augmentation that's added variance to the training data as well as supplying the model with more data to learn. All experiments were done with only one varying property between one another to ensure the changes that are seen can be linked to the single different property with confidence. The trials proved two things: data normalization and augmentation had a great positive impact on the performance, the proposed modified architecture with removed dropout layers did in fact deliver good results. In terms of possibility for future improvements there are a couple of directions - For starters, there is potential in experimenting with different normalization techniques, only one was considered in this article, but there are few possibilities that were briefly explored during the writing of the article - such as Per-channel energy normalization, BoxCox normalization, etc. Another direction is making the system segment an audio file into chords and not only classify whole sample as one chord, another possibility can be doing ACR in real-time - making the system react like an online algorithm to the input.

## 8. References

2021. *Audio Chord Estimation task in MIREX site*.
2023. *Apple Music - streaming platform*.
2023. *Easy and intuitive - Simply Piano*.
2023. *Shazam - Music Discovery, Charts & Song Lyrics*.
2023. *SoundHound — Technology for a voice-enabled world*.
2023. *Spotify - Web Player: Music for everyone*.
2023. *VERSILIAN STUDIOS CHAMBER ORCHESTRA 2*.
2023. *Yousician - Learn Guitar, Piano, Ukulele*.
- Abayomi-Alli, Olusola, Damaševičius, Robertas, Qazi, Atika, Adedoyin-Olowe, Mariam, & Misra, Sanjay. 2022. Data Augmentation and Deep Learning Methods in Sound Classification: A Systematic Review. *Electronics*, **11**(11), 3795.
- Agarap, Abien Fred. 2019. *Deep Learning using Rectified Linear Units (ReLU)*.
- Boulanger-Lewandowski, Nicolas, Bengio, Yoshua, & Vincent, Pascal. 2013. Audio Chord Recognition with Recurrent Neural Networks. *In: International Society for Music Information Retrieval Conference*.
- Burgoyne, John Ashley, Pugin, Laurent, Kereliuk, Corey, & Fujinaga, Ichiro. 2007. A cross-validated study of modelling strategies for automatic chord recognition in audio.
- Chai, Wei, & Vercoe, Barry. 2005 (01). Detection of Key Change in Classical Piano Music.

- Chen, Tsung-Ping, & Su, Li. 2019. Harmony Transformer: Incorporating Chord Segmentation into Harmony Recognition. In: *International Society for Music Information Retrieval Conference*.
- Cho, Taemin. 2014 (01). *Improved techniques for automatic chord recognition from music audio signals*. Ph.D. thesis.
- Cho, Taemin, & Bello, Juan. 2011 (01). A Feature Smoothing Method for Chord Recognition Using Recurrence Plots.
- Cho, Taemin, & Bello, Juan P. 2014. On the relative importance of individual components of chord recognition systems.
- Choi, Keunwoo, Fazekas, György, Cho, Kyunghyun, & Sandler, Mark. 2021. *A Comparison of Audio Signal Preprocessing Methods for Deep Neural Networks on Music Tagging*.
- Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, & Toutanova, Kristina. 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*.
- Euler, Leonard. 1739. *Tentamen novae theoriae musicae ex certissimis harmoniae principiis dilucide expositae*.
- Fujishima, Takuya. 1999. Realtime Chord Recognition of Musical Sound: a System Using Common Lisp Music.
- Huang, Cheng-Zhi Anna, Vaswani, Ashish, Uszkoreit, Jakob, Shazeer, Noam, Simon, Ian, Hawthorne, Curtis, Dai, Andrew M., Hoffman, Matthew D., Dinculescu, Monica, & Eck, Douglas. 2018. *Music Transformer*.
- Humphrey, Eric J., & Bello, Juan P. 2012. Rethinking Automatic Chord Recognition with Convolutional Neural Networks.
- Humphrey, Eric J., & Bello, Juan Pablo. 2015. Four Timely Insights on Automatic Chord Estimation. In: *International Society for Music Information Retrieval Conference*.
- Humphrey, Eric J., Cho, Taemin, & Bello, Juan P. 2012. Learning a robust Tonnetz-space transform for automatic chord recognition. Pages 453–456 of: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.
- Jadhav, Yogesh, Patel, Ashish, Jhaveri, Rutvij H., & Raut, Roshani. 2022. Transfer Learning for Audio Waveform to Guitar Chord Spectrograms Using the Convolution Neural Network.
- Jeni, László, Cohn, Jeffrey, & De la Torre, Fernando. 2013 (09). Facing Imbalanced Data - Recommendations for the Use of Performance Metrics. vol. 2013.
- Jones, Steve. 2000. Music and the Internet. *Popular Music*, **19**(2), 217–230.
- Kingma, Diederik P., & Ba, Jimmy. 2017. *Adam: A Method for Stochastic Optimization*.
- Korzeniowski, Filip, & Widmer, Gerhard. 2016a. *Feature Learning for Chord Recognition: The Deep Chroma Extractor*.
- Korzeniowski, Filip, & Widmer, Gerhard. 2016b. A fully convolutional deep auditory model for musical chord recognition.
- Korzeniowski, Filip, & Widmer, Gerhard. 2018. *Improved Chord Recognition by Combining Duration and Harmonic Language Models*.
- Korzeniowski, Filip, Sears, David R. W., & Widmer, Gerhard. 2018. *A Large-Scale Study of Language Models for Chord Prediction*.
- Madhu, Aswathy, & Kumaraswamy, Suresh. 2019. Data Augmentation Using Generative Adversarial Network for Environmental Sound Classification. Pages 1–5 of: *2019 27th European Signal Processing Conference (EUSIPCO)*.
- Martinez, Manuel, & Stiefelhofen, Rainer. 2018. *Taming the Cross Entropy Loss*.
- Mauch, Matthias, & Dixon, Simon. 2010. Simultaneous estimation of chords and musical context from audio.
- McCain, Abby. 2023. *How much is the music industry worth?*
- Nadar, Christon, Abeßer, Jakob, & Grollmisch, Sascha. 2019 (04). Towards CNN-based Acoustic Modeling of Seventh Chords for Automatic Chord Recognition.
- Nakayama, Shota, & Arai, Shuichi. 2018. DNN-LSTM-CRF Model for Automatic Audio Chord Recognition. Page 82–88 of: *Proceedings of the International Conference on Pattern Recognition and Artificial Intelligence*. PRAI 2018. New York, NY, USA: Association for Computing Machinery.
- Osmalsky, Julien, Embrechts, Jean-Jacques, Van Droogenbroeck, Marc, & Pierard, Sébastien. May 2012. Neural networks for musical chords recognition. In: *Journées d'informatique musicale*.
- O'Hanlon, Ken, & Sandler, Mark B. 2021. FifthNet: Structured Compact Neural Networks for Automatic Chord Recognition.
- Papadopoulos, H'el'ene, & Peeters, Geoffroy. 2011. Joint estimation of chords and downbeats from an audio signal.
- Park, Jonggwon, Choi, Kyoyun, Jeon, Sungwook, Kim, Dokyun, & Park, Jonghun. 2019. *A Bi-directional Transformer for Musical Chord Recognition*.
- Pauwels, Johan, & Martens, Jean-Pierre. 2014. Combining musicological knowledge about chords and keys in a simultaneous chord and local key estimation system.
- Salamon, Justin, & Bello, Juan Pablo. 2017. Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification. *IEEE Signal Processing Letters*, **24**(3), 279–283.
- Shah, Ayush Kumar. 2020. *Guitar-Chords-recognition*.
- Sheh, Alexander, & Ellis, Daniel. 2003. Chord Segmentation and Recognition using EM-Trained Hidden Markov Models. 11.
- Sigtia, Siddharth, Boulanger-Lewandowski, Nicolas, & Dixon, Simon. 2015. Audio Chord Recognition with a Hybrid Recurrent Neural Network. In: *International Society for Music Information Retrieval Conference*.
- Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, & Salakhutdinov, Ruslan. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, **15**(56), 1929–1958.
- Sumi, Kouhei, Itoyama, Katsutoshi, Yoshii, Kazuyoshi, Komatani, Kazunori, Ogata, Tetsuya, & Okuno, Hiroshi G. 2008. Automatic chord recognition based on probabilistic integration of chord transition and bass pitch estimation.
- Taenzer, Michael, Abeßer, Jakob, Mimiakis, Stylianos I., Weiss, Christof, Müller, Meinard, & Lukashevich, Hanna. 2019. *Investigating CNN-based Instrument Family Recognition for Western Classical Music Recordings*.
- Ueda, Yushi, Uchiyama, Yuki, Nishimoto, Takuya, Ono, Nobutaka, & Sagayama, Shigeki. 2010 (04). HMM-based approach for automatic chord detection using refined acoustic features.
- Ulrich, John Wade. 1977. The Analysis and Synthesis of Jazz by Computer. In: *International Joint Conference on Artificial Intelligence*.
- Wei, Shengyun, Zou, Shun, Liao, Feifan, & weimin lang. 2020. A Comparison on Data Augmentation Methods Based on Deep Learning for Audio Classification. *Journal of Physics: Conference Series*, **1453**(1), 012085.
- Weller, Adrien, & Ellis, Daniel. 2003. Structured prediction models for chord transcription of music audio.
- Wikipedia contributors. 2023a. *Chord (music)* — Wikipedia, The Free Encyclopedia. [Online; accessed 23-May-2023].
- Wikipedia contributors. 2023b. *Equal temperament* — Wikipedia, The Free Encyclopedia. [Online; accessed 3-June-2023].
- Wikipedia contributors. 2023c. *Key (music)* — Wikipedia, The Free Encyclopedia. [Online; accessed 3-June-2023].
- Wikipedia contributors. 2023d. *MIDI* — Wikipedia, The Free Encyclopedia. [Online; accessed 23-May-2023].
- Wikipedia contributors. 2023e. *Music industry* — Wikipedia, The Free Encyclopedia. [Online; accessed 23-May-2023].
- Wikipedia contributors. 2023f. *Music theory* — Wikipedia, The Free Encyclopedia. [Online; accessed 23-May-2023].

Wikipedia contributors. 2023g. *Octave* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 3-June-2023].

Wu, Yiming, & Li, Wei. 2019. Automatic Audio Chord Recognition With MIDI-Trained Deep Feature and BLSTM-CRF Sequence Decoding Model. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, **27**(2), 355–366.

Yamashita, Rikiya, Nishio, Mizuho, Do, Richard Kinh Gian, & Togashi, Kaori. 2018. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, **9**(4), 611–629.

Zhou, Xinquan, & Lerch, Alexander. 2015. CHORD DETECTION USING DEEP LEARNING. *ISMIR*.