Real-time pitch detection using resource constrained IoT Device

Job Römer, University of Twente, The Netherlands

ABSTRACT

A trained ear for the recognition of notes, intervals and chords in music is a difficult skill for beginning musicians to develop due to lack of reference of these musical aspects and boring nature of its training. A system or program could rectify this issue and allow users to train their ears for this purpose. Multiple existing algorithms can distinguish pitch and chords from given audio files, using signal analysis or neural networks. However, these solutions provide only the recognition of one of the three relevant pitch based aspects. This research contributes to the development of a system that does all three by using models based on convolutional neural networks. New, custom, datasets were created to facilitate training and evaluation of the models as well. The final models were optimized and vary in terms of accuracy but show promise to be further developed into a reliable system.

Additional Key Words and Phrases: Pitch Recognition, Real-Time Pitch Recognition, IoT device, convolutional neural network, Mel-Filterbank.

1 INTRODUCTION

1.1 Context and Motivation

Like many other skills, learning to play a musical instrument is difficult and time-consuming. Through a combination of physical movements and coordination, music theory, “an ear” for pitch and “a feeling” for rhythm, a musician can express themselves in nearly infinite ways.

While the former two skills can be practiced and taught, the latter two cannot be simply explained through lecture or learned through practice of an instrument. These are some of the most vital skills one can have as a musician, as reinforced by (Willis, 1998) [1] in his book, and may be what constitutes intuition in music. However, training the ear for these purposes requires time, dedication and reference of different pitches and rhythms and can often be boring or unengaging as noted by (Rizqyawan, & Hermawan, 2015) [2]. Usually, one is exposed to exercises through a music teacher and only after the student has become accustomed to their instrument of choice. Due to the boring nature of ear training relative to the practice of their instrument, many inexperienced students choose to skip it. However, they later realize that they cannot play new songs because they do not know patterns that they had previously blindly copied for other songs and cannot create their own solos, songs or their “own sound”. A trained ear can rectify this, and to this end, this research will explore a way to help people train their ears, without the need for a music teacher’s continued effort.

TScIT 37, July 8, 2022, Enschede, The Netherlands

1.2 Research

This research puts the focus exclusively on the problem of pitch recognition, specifically in terms of single notes, intervals, and simple chords, and details the development of a prototype of a recognition system run on an IoT device. This tool should be able to listen to real-time input in the form of musical data originating from a musician playing their instrument and provide feedback about the note, interval, or chord that they have played.

The goal of this research is to determine the feasibility and potential of such a system. The research questions following from this goal are:

1. What are available methods to detect one or more pitches and how accurate and applicable to the proposed system are these methods?
2. In what ways can we optimize the selected pitch recognition method’s accuracy?
3. How does the realized system perform in terms of accuracy?

1.3 Approach and Structure

As stated, the chosen approach to answering these questions is the development of a prototype of the proposed pitch recognition system. A literary review of the state of the art and older research answers the first research question and additionally serves the secondary purpose of laying the groundwork for the selection of the pitch recognition method for the proposed system. This literary review is detailed in ***Section 2***. The selected method is then implemented. The context surrounding this method and the structure and workings of the implementation are detailed in ***Sections 3 and 4***, respectively. The latter section also describes parameters that were tested in order to answer the second research question. Finally, the system is evaluated in a computer environment with given audio files and on an embedded device with real time playing, which present the answers to the final research question. The results and analysis of these experiments are detailed in ***Section 5***.

1.4 Definitions and Terminology

In order to avoid confusion, a few terms and definitions are explicitly mentioned in this section. Table 1 contains the important music theory related definitions and terminology used within this paper.

Table 1. Definitions and terminology for this paper.

Term	Explanation
Pitch	A frequency that can be related to a musical note. The term note and pitch will be used interchangeably in this paper.
Sharps/Flats	Some notes are enharmonically equivalent, meaning that they produce almost the same frequency. The most important are sharps and flats, detailed in Table 2 in Appendix A. All flat notes will be referred to as their enharmonic sharp equivalents to avoid unnecessary complexity of the paper.
Interval	Two notes played at the same time. The name of the interval is determined by the distance between the two notes. Appendix A contains more information on intervals and interval inversion.
Chord/Triad	Three or more notes played at the same time. Both terms will be used to refer to simple chords that consist exactly of three notes. These two terms are therefore used interchangeably in this paper and cover only a small subset of possible chords.

2 LITERARY REVIEW

In this section, multiple methods that may be used or adapted to recognize notes, intervals or chords are highlighted and some requirements for the proposed system are identified.

This research falls under the Music Information Retrieval (MIR) research area, which is concerned with retrieving information like rhythms, time signatures, keys, instruments, notes, and other musical aspects from written music scores or from audio input, which includes existing audio files as well as real-time audio (Jensen & Andersen, 2004) [3]. Many researchers have already researched ways to extract pitches from single notes, notes that make up a chord and even the notes of all simultaneously playing instruments. All of them use techniques to compute time-frequency representations of the audio, which is often referred to as a Spectrogram. Algorithms can then perform feature extraction using this spectrogram to determine frequencies, instruments and other quirks present in the musical data. Multiple approaches for feature extraction exist and one of the more prominent in the present day is neural networks.

(Voinov et al., 2019) [4] analyze several algorithms for single pitch detection and propose a three-layer neural network implementation for chord recognition that has an accuracy of 95-99% on a small set of 10 simple chords. They also note that, for recognizing multiple notes at the same time (intervals and chords), single note recognition algorithms are too inefficient and therefore neural networks should be the optimal choice for this kind of task. Similarly, (Korzeniowski et al., 2015) [5] use convolutional neural networks to achieve a chord recognition model that allegedly performs on par with or better than state-of-the-art competitors (with an accuracy of 82.9%) on the set of all major and minor triads. (Sigtia et al., 2016) [6] put this approach in a slightly different context by using the neural network for the purposes of polyphonic piano transcription. Analogous to a speech recognition system, two neural networks were used: one acoustic model for estimating probabilities in pitch and a music language model which correlates intervals and chords over time. The end result is a model that can transcribe polyphonic piano music into a written score. With adaptation, this model can be efficient enough to run in real-time applications.

Older research, dating back to 2009 and prior, also consider pitch recognition methods in the form of Signal Processing algorithms using different principles and spectrogram techniques. (Black & Donohue, 2000) [7] mention and evaluate multiple of these algorithms, those being a generalized spectrum-based algorithm, an autocorrelation-based algorithm, and a cepstrum-based algorithm. (Huang & Yu, 2018) [8] make use of the cepstrum-based algorithm to detect stable pitches in humming melodies and (Kuhn, 1990) [9] provides a technical explanation of the workings and drawbacks of autocorrelation in real-time applications, one of the latter being that it is a computationally intensive method. Kuhn also proposes another method: the FMP method, which is detailed in his journal article.

Lastly, probabilistic models in conjunction with adaptive pattern matching are used by (Kashino & Murase, 1998) [10] to identify notes and the instruments that produce them.

These solutions are all interesting and have their pros and cons but are all focused on either single note recognition or chord recognition. The method for my proposed system has to have the capability of recognizing single notes, intervals, and chords rather than only one of these three. Moreover, the chosen method needs to be capable of recognizing audio from a musician's playing in real-time and doing so efficiently to make it viable for an IoT device as well. Specific challenges result from differing audio qualities, possible limitations in terms of storage, battery life and computing power of the IoT device and real-time recognition.

3 PROPOSED METHOD

The following section covers the choice of pitch recognition method and in general how the chosen solution operates. In the first subsection, the chosen method is revealed and the second will focus on a brief explanation of the workings of this method.

3.1 Choice of Method

The selection of the proposed method is carefully considered using applicability to the problem, speed, efficiency, and accuracy as the general criteria. Due to the first criteria, any simple signal processing algorithms such as autocorrelation are unfit methods for the system due to their inability to detect multiple notes whereas models based on convolutional networks, similar to (Korzeniowski et al., 2015) [5]'s work show

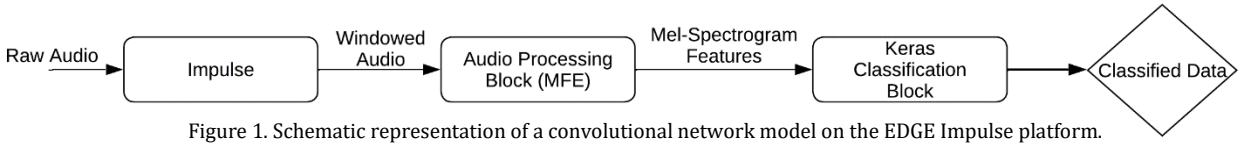

Figure 1. Schematic representation of a convolutional network model on the EDGE Impulse platform.

more promise, since these neural networks are not only efficient and fast, but also show the capability of recognizing multiple pitches at once with adequate accuracy. Moreover, this solution shows to be viable for a real-time applications as well, as shown by (Sigtia et al., 2016) [6]. Therefore, convolutional neural networks are used as the chosen pitch recognition method for the final system.

3.2 Convolutional Neural Networks

A neural network is an algorithm loosely modelled after the human brain, consisting of neuron layers that can find relationships between labelled data and the features they have. In the end, the neural network will be able to classify new data based on features it learns from previous data and the relationships between this data and the corresponding labels. As (Korzeniowski et al., 2015) [5] explain in their paper, convolutional neural networks differ from traditional neural networks in that they generally make use of two additional types of computation layers, those being *convolution layers* and *pooling layers*. A convolution layer computes a convolution of its input to find out how the features affect each other. A pooling layer generalizes the features from the convolution layers, helping the network learn faster due to the reduced number of features and parameters resulting from the generalization. The combination of efficiency, accuracy and applicability to the problem made this method perfect for the proposed system.

4 IMPLEMENTATION

In this section, the development of the proposed system is detailed. The first subsection covers the general structure of the models and their purpose, while the second provides in-depth explanations of the separate parts that make up the models, such as the preprocessing and the convolutional neural network.

4.1 Model Design

The system prototype consists of 4 different models:

- The **Single Note** model classifies instances of one pitch. In total, there are 12 notes commonly found in Western music and these notes were used as the 12 classes that the model would have to distinguish between. Along with a "No Note" class, the model consists of **13 total classes**.
- The **Interval** model classifies instances of intervals based on distance between notes. There are 12 different intervals that one can construct using the notes in Western Music and as such this model has **13 classes**, the last one being "No Interval".
- The **Interval Root** model classifies instances of intervals based on the lower (root) note of the interval. Together with the Interval model, the outputs can be used to exactly determine what interval and what exact notes are present in an audio signal. In this research, this combination is not implemented but could be added in future work. Analogous to the Single Note model, there are **13** distinguishable **classes** for this model.
- The **Chord** model classifies instances of simple major or minor chords. A restriction to a reasonable set of classes is made because the total number of different chords is too large to otherwise consider. There are exactly 12 major chords and 12 minor chords one can construct with the notes commonly found in Western music. Along with the "No Chord" class, this model thus consists of **25 classes**.

4.2 Neural Network Design

The EDGE Impulse Website facilitates creation, management, training, and testing of the convolutional models. The platform moreover makes deploying the models for an IoT device simple as well. The IoT device of choice is a Samsung Galaxy A51 phone in this research but more devices, like a Raspberry Pi 4, are also supported by the platform.

A model based on a convolutional network has four parts in this implementation:

- A dataset, which is split into a training set and a testing set (typically with a ratio of 80/20)
- A time series data block
- An audio processing block (MFE) and
- A (Keras) classification block.

This general structure of the models is schematically depicted in Figure 1 and the next subsections will go over each of these parts and explain what was done, how each block works and the parameters they have.

4.2.1 Dataset

The datasets to train the models are completely custom and all datapoints are created by me. These datapoints are all created using the Ableton Live 11 Digital Audio Workstation (DAW) and 9 different MIDI and free VST instruments that come packaged with the DAW or can be found online. All data is recorded at 120 BPM and each note, interval or chord is held for 2 bars, resulting in 2 seconds of audio for each instance.

The recorded data is split into a total of three different datasets: one for single notes, one for intervals, which the two interval models share, and one for chords. The aim is to have sufficient data that represents all possible classes for all selected instruments in all three datasets and as such the single note dataset consists of 289 instances, the interval dataset consists of 1009 instances and the chord dataset consists of 984 instances. More data would be needed to make these datasets represent all possible audio instances, but these datasets should cover most common instances and provide a nice basis for the research.

4.2.2 Time Series Data Block

In this block, instances of audio are divided into windows in order to create more datapoints. Parameters include length of the windows, window increase (in case a sample is larger than the window, a sliding window will go over the sample. This parameter tells the block by how much to increase the sliding window each time), an option of zero-padding the data (in case a sample is shorter than the window, silence is added at the end), and sampling frequency (which determines audio quality after processing). This windowed data is then fed into the audio processing block.

4.2.3 Mel Frequency Energy Processing Block

This block transforms the windowed audio input into a spectrogram and features for the classification block. The selected Mel-Filterbank Energy (MFE) block, like its name suggests, uses a Mel-filterbank to extract features from a spectrogram created by a Fast Fourier Transform (FFT). This filterbank excels at audio that can be distinguished by human ears, making it very applicable for distinguishing pitches.

A Mel-filterbank is a collection of triangular filters which is based on human perception experiments as noted by (Tak, Agrawal, & Patil, 2017) [11] who used the Mel Filterbank Cepstral Coefficient (MFCC) technique for automatic speech recognition. To mirror the human ears, these filterbanks are designed to extract more features with lower frequencies and less features with high frequencies. This results in a more refined spectrogram called a Mel Spectrogram.

The MFE block has a few parameters concerning the FFT, including frame length (the length of each frame in seconds), frame stride (the step between successive frames) and FFT Length (the number of FFT points), as well as a few parameters for the Mel-filterbank including filter number (how many triangular filters are used), low frequency (lower bound for frequencies in the audio) and high frequency (upper bound for frequencies in the audio). There is also a normalization step that is performed afterward in the form of a noise floor parameter, which disregards any audio data under a specified decibel level. After processing, the data is fed into the classification block.

4.2.4 Keras Classification Block

After feature extraction, the features are given to the Keras Classification block for classification of the windows. This block makes use of the convolutional neural networks, the general workings of which are explained in Section 3.2. The training data is further split into an actual training set and validation set, the former of which is used to train the model and the latter of which is used to tune parameters of the classifier. Additionally, the EDGE Impulse platform makes use of the validation set to create a feature explorer and confusion matrix to illustrate how the network classifies data points in this dataset. This is useful in identifying potential reasons for low accuracies in models.

Parameters of this block include number of training cycles (number of epochs to train the network on), learning rate (how fast the neural network learns), validation set size (changes ratio for training and validation data during training. The default ratio is 80/20) and options to enable automatic balancing of the dataset for underrepresented classes and data augmentation in the form of adding noise or masking time and frequency bands. The block also provides control over the structure of the network. One can add more layers and change parameters of these layers as well. One final, parameter that is important for this block is the confidence threshold, which determines how confident a model needs to be before a classification can be made. If the confidence value for one class exceeds the confidence threshold, the network will classify the sample as the class with the highest confidence value. After this block finishes, one has a classified data point.

4.3 Optimization of parameters

The parameters, specifically those of the Keras Classification block, present interesting opportunities to improve the accuracy of the models. The number of training cycles and learning rate can be used to increase how well the model learns from instances in the training set, and naturally should be as high as possible while avoiding potential overfitting of the models. Overfitting refers to the phenomenon of a network or statistical model performing worse on unseen data because it learns too many details of its training data. Additionally, the confidence threshold may have effect on the accuracy of the models in the sense that considering lower confidence values may make it easier for a network to classify instances. Finally, the network structure may have impact on how well the model performs. Deciding between a number of convolution layers, kernel size, number of filters and overall structure of the network all affect the accuracy. Some of these parameters are experimented with and these experiments and their results are detailed in the next section.

Parameters of the time series data block and MFE block may induce changes in accuracy as well, however these parameters are harder to change in the model's favor as these changes directly affect the output audio or spectrogram. As a result, these changes may specifically favor accuracy on certain data points while hurting the accuracy on others. An obvious example of this in the MFE block is the noise floor parameter. If this parameter is set too high, data points of high volume may benefit from a

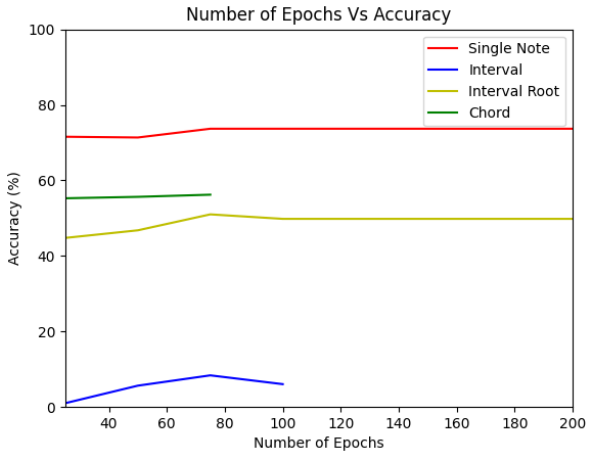

Figure 2. Graph plotting accuracy against numbers of epochs

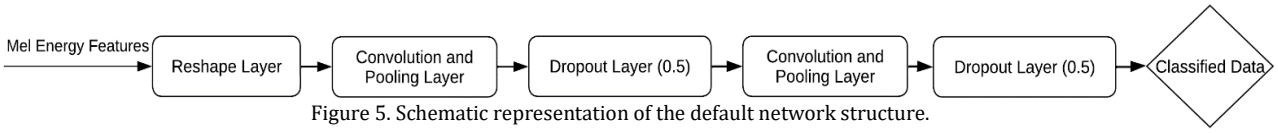

Figure 5. Schematic representation of the default network structure.

lower amount of unwanted noise while the frequencies in low volume data points may entirely be disregarded, since they might fall under the noise floor.

5 EXPERIMENTS AND RESULTS

In the first subsection, results of experiments regarding the effects of four parameters on the model accuracies are presented, in the second subsection, possible explanations for low accuracies of certain models are provided and in the final subsection, the final experiment is detailed, in which the models are run on an embedded IoT device and accuracies of the models in two different environments are compared.

5.1 Parameter Experiments

The aim of this section is to raise the accuracy of the models and the effects of the parameters are tested through systematic experiments. The tested parameters are the number of epochs, learning rate, confidence threshold and structure of the networks. The models are evaluated on the accuracy that they achieve on the datapoints in the testing dataset.

5.1.1 Number of Epochs

As can be seen in Figure 2, the number of epochs that the model goes through seems to have relatively little effect on the accuracy. At low numbers of epochs, the accuracy varies around 2% to 7%, while converging at higher numbers of epochs. This also suggests that the models are not overfitting since there is no perceived drop in accuracy at these higher values. Due to constraints on the runtime of models imposed by the EDGE Impulse platform, some datapoints are not obtainable, which is why some models lack tests after a certain number of epochs. This may be rectified in future work if one has access to an unrestrained model.

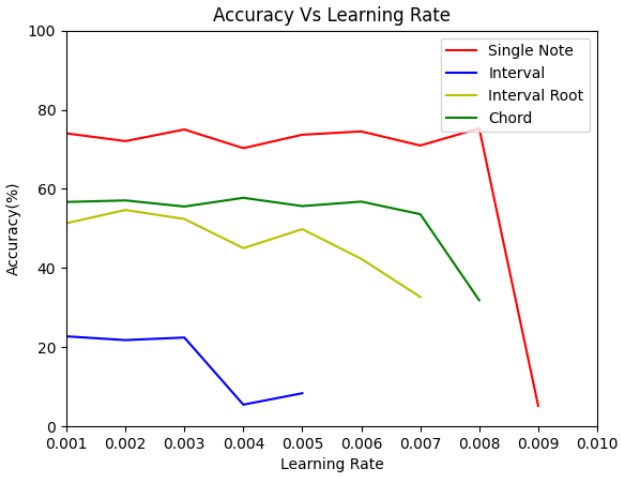

Figure 3. Graph plotting accuracy against the learning rate.

5.1.2 Learning Rate

Learning rate seems to have more of an impact on accuracy, as shown in Figure 3. The accuracies seem to vary with different learning rates, between 2% and 12% to be exact, but a clear correlation between higher and lower learning rates cannot be made, outside of the fact that lower learning rates seem to perform slightly better than higher learning rates in most cases. It is also worth noting that with very high learning rates, the accuracies drop significantly, likely because these higher values cause the models to overfit. As was the case with the epochs experiment, some datapoints are unobtainable due to time constraints imposed by the EDGE Impulse platform.

5.1.3 Confidence Threshold

The confidence threshold seems to have a more significant effect on the accuracy of the models, as can be seen in Figure 4. Changing this parameter induces large variations of potentially 30% for the Interval and Chord models, 15% for the Interval Root model and a less significant 8% for the Single Note model. In general, confidence thresholds between 0.2 and 0.4 produce maximum accuracies, and lower or higher confidence thresholds cause the accuracy to drop significantly, with a confidence value of 1 causing an accuracy of 0% in all models. This suggests that the models have a hard time pinpointing the exact classes instances belong to, seeing as a lot of data instances are predictably classified as the "Uncertain" class with higher confidence thresholds.

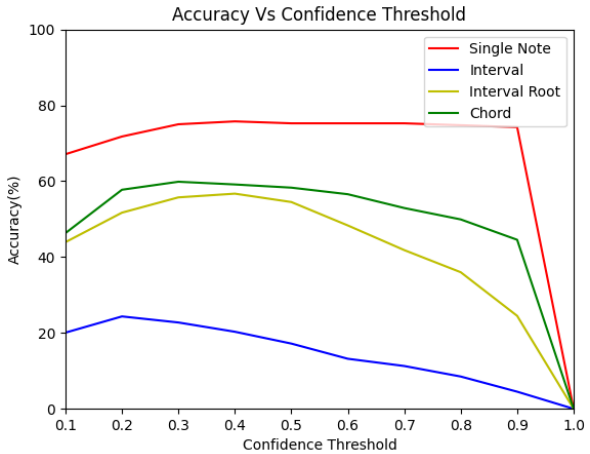

Figure 4. Graph plotting accuracy against the confidence threshold

5.1.4 Network Structure

This experiment consists of only a small subset of possible network structures due to time constraints. The general structure that is used is schematically depicted in Figure 5. The layers that are most relevant to this experiment are the sets of convolution and pooling layers. EDGE Impulse allows the user to

change the number of convolution layers in each set, up to a maximum of 3.

The representation of "x-y" is used to denote the network structure on the x-axis, where x is the number of convolution layers in the first set and y the number of convolution layers in the second set.

Figure 6 shows that all models perform better with relatively simple network structures. It also makes apparent that network structure affects accuracy only marginally, showing only small variations in accuracy, which range from only 2% between two relatively close network structures to at most 15% between relatively distant network structures. Of note as well is that the "3-3" datapoint for the Chord model is unobtained due to time constraints imposed by the EDGE Impulse platform.

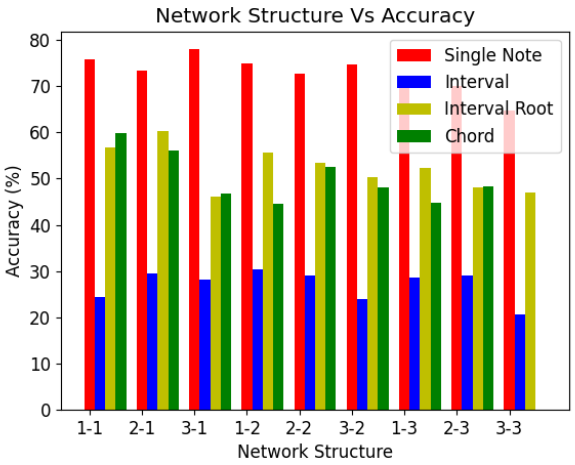

Figure 6. Graph plotting accuracy against the network structure.

The Single Note model shows preference for a structure with 3 convolution layers followed by 1 convolution layer, beating the next best option by about 2%. The Interval model shows peak accuracy with 1 convolution layer followed by 2 convolution layers, beating the next best option by 0.5%. The Interval Root and Chords models perform best with a structure containing two instances of 1 convolution layer, beating the next best option by around 3% and 4% respectively.

5.2 Explanations for low accuracies

The Single note model performs the best by far, but the other models are somewhat lacklustre in their observed accuracy. There are likely many reasons why their accuracy is lower and, since we cannot accurately predict how the neural network itself works, we can only make reasonable assumptions to explain why the network may be mislabelling instances. In this subsection, three possible causes for low accuracy of the models in this environment are mentioned.

5.2.1 Challenges due to interval inversion

The confusion matrix of the Interval model reveals that the model most often mis qualifies windows as an interval that is relatively close to the actual interval (major 2nd vs minor 2nd for example) or as one that is or is close to the ***inverted interval*** associated with it. Inverting an interval means moving one of the notes in it up or down an octave, which preserves the harmonic relationship between the notes but increases/decreases the distance between them and garners the interval a new name. Figure 8 in Appendix A illustrates this concept of inversion. The model may have trouble differentiating intervals and their inversions and may misclassify one as the other more often, which would provide one reason for its lacklustre accuracy.

5.2.2 Possible bias toward instrument timbre

All models, except for the Single Note model, show another strange behaviour. The models group sample windows of the validation set not on the perceived frequencies but rather on ***instrument timbre***, which refers to the sound indicative of a particular instrument. A violin and a trumpet have a different timbre for example, and the models seem to group windows in the validation set primarily based on the texture of the instrument's sound (grouping violins and trumpets separately). This leads me to believe that the features extracted by the audio processing block are more geared to differentiating one sound from another based on its "texture" or timbre, rather than frequencies. This may be rectified in the future by writing a custom audio processing block that applies a higher weight to perceived frequencies, as the EDGE Impulse provides little to no possibilities of interacting with output features of the MFE block.

5.2.3 Multiple frequencies clutter the signal

This issue mostly applies to the Interval Root model specifically but may also be accountable in the Interval and Chord models. Since the Single Note recognition model performs well, it might be assumed that recognising the root note of an interval should perform about equally in terms of accuracy, as it analogously should only have to recognise one note. However, it is also fairly obvious that the introduction of a second note may complicate the labelling process due to overlapping frequencies and overtones produced by the second note. Additionally, this model may struggle finding which of the two notes is the root note of the interval since it perceives the frequencies of both notes.

5.3 Deployment of models on IoT Device

The final test for each model consist of a few classifications using the selected IoT Device (which is a Samsung Galaxy A51 smartphone in this research) and a comparison of their accuracy between this device and the computer environment in which the models were trained.

5.3.1 Test outline

Each test dataset consists of 30 new custom instances (created by me) that the models have not encountered before. The models are first started and then an appropriate instance of a note, interval or chord is played on a musical instrument. The first 3 or more consecutive windows of said instance that are labelled as the same class will be taken as the model's classification and when less than 3 windows of the instance are labelled as a certain class or the model classifies the instance as the "Uncertain" class, the result of the test is noted as an ***uncertain*** in the results. Afterward, the number of correctly identified instances for each model and the corresponding accuracy of the

model are calculated. An uncertain is treated as a wrong answer for this calculation. Additionally, more calculations are performed to determine accuracy on some particular groups of instruments per model.

A wide range of lower and higher instances (in terms of frequency) as well as instances using instruments that the models are not trained on are included to test the robustness of the models. This is to ensure that the models are evaluated on as many possible instances and to identify potential weak points of the models that are not easily verifiable in a computer environment.

Of note is that the microphone on the IoT device produces in audio quality reduced data points when working in conjunction with the models as opposed to the microphone's audio quality with the voice recorder app on the phone. This is one of the secondary reasons to test the models when embedded on the device; evaluating how robust the models are even with subpar recording equipment.

5.3.2 Single Note Model

This test consists of 12 piano instances, 6 acoustic guitar instances, 3 trumpet instances, 3 organ instances, and one instance of a violin, square wave synthesiser, kalimba, saxophone, accordion, and the human voice.

In total, the Single Note model correctly classifies 21 out of 30 instances, resulting in an **accuracy of 70%**. Moreover, the accuracy on the piano instances is around 67%, the guitar instances are classified with 50% accuracy, the trumpet instances are classified with around 67% and the church organ instances are correctly classified with 100% accuracy.

However, low notes seem to be mislabelled most often in this test and this is likely because there are few of these lower notes in the training dataset.

5.3.3 Interval Model

This test contains 12 piano instances, 6 acoustic guitar instances, 3 sawtooth wave synthesiser instances, 3 strings instances, and one instance of a trumpet, an electric Rhodes piano, clarinet, violin, electric organ, and a church organ.

In total, the Interval model correctly classifies 5 out of 30 instances, resulting in an **accuracy of around 17%**. Moreover, the accuracy on the piano instances is 25%, the guitar instances are classified with around 17% accuracy, and the sawtooth synth and strings instances are classified with 0% accuracy.

One issue with this model in particular seems to be that the model has difficulties classifying the same interval when produced by different instruments or played in different octaves. This may suggest that the data was not representative enough in this model, seeing as instances are correctly identified in some cases, but not in others.

5.3.4 Interval Root Model

This test consists of 12 piano instances, 6 acoustic guitar instances, 3 electric organ instances, 3 trumpet instances, and one instance of an accordion, clarinet, square wave synthesiser, strings, electric Rhodes piano and a kalimba.

In total, the Interval Root model correctly classifies 10 out of 30 instances, resulting in an **accuracy of around 33%**. Moreover, the accuracies on the piano and guitar instances are both 33.33% and around 67% of the electric organ and trumpet instances are correctly classified. These four sets of instances comprise the entire set of correctly classified instances as well, which reflect the model's low accuracy.

However, the results of this test also reveal that the model often (correctly) identifies the top note of the interval, rather than the root. For reverse engineering of the interval, it is only important that one of the two pitches is correctly identified, but the intention is for this network to identify the root note of the interval, which in the majority of cases does not happen. Future work may instead focus on a model that identifies the top note of an interval, rather than the root note. Tests that pit these two different models against each other should also be performed to determine which of these two methods is more reliable.

5.3.5 Chord Model

This test contains 12 piano instances, 6 acoustic guitar instances, 3 strings instances, 3 electric Rhodes piano instances, and one instance of a church organ, concert flute, trumpet, square wave synthesiser, saxophone, and a French horn.

In total, the Chord model correctly classifies 14 out of 30 instances, resulting in an **accuracy of around 47%**. Moreover, the accuracy on the piano instances is around 58%, the accuracy on the guitar instances is 0%, and around 67% of the strings and electric Rhodes piano instances are correctly classified.

Lower accuracy on the guitar instances may be caused by underrepresentation of this instrument. The guitar's strings keep resonating after they have been played in *quick succession*, and the notes form a chord together. The model is exclusively trained on chords in which the chord's notes are always played *at the exact same time*, however. The model is also not familiar with the sound of a guitar either (although that is one of the reasons to include the instrument in the test) and these oversights may likely cause the model to have poor accuracy on these specific instances.

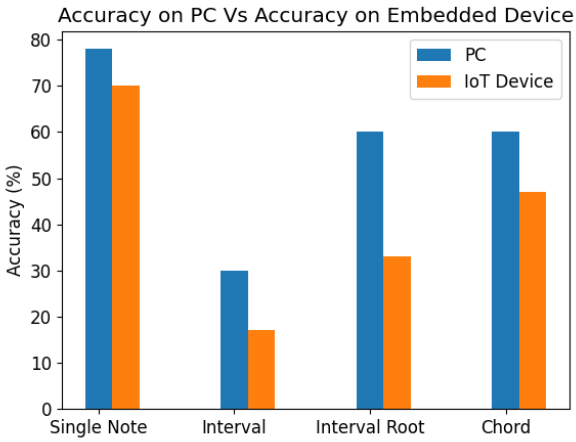

Figure 7. Graph plotting the accuracy in a PC environment and in an environment of an Embedded Device

5.3.6. Comparison between tested environments

As can be seen in Figure 7, the models perform noticeably worse on the embedded device. In some cases, the gap in accuracy may be as large as 27% while in others the difference in accuracy is closer to around 10%. This may be caused by aforementioned microphone issues, reduced audio quality in general, background noise that is too prominent during the test or any of the previously mentioned issues some models may have.

6 CONCLUSION

6.1 General Conclusion

The chosen convolutional network based models show promise to be reliable recognition models in the future, but more research should be done to further develop and optimize them. More resources, such as a custom processing block, more elaborate models that can be run for longer and more data, should be prioritized to make this type of system reliable and warrant future use.

6.2 Conclusion Research Question 1

The current state of the art includes several methods, which work well for their intended purpose. The most promising for this research concerns convolutional neural networks due to its applicability to the problem and potential to be viable for a real-time system run on an IoT Device.

6.3 Conclusion Research Question 2

Several parameters, affect accuracy in different ways and four of these parameters are tested:

- The number of epochs seems to have a relatively little effect on accuracy. Generally, more training cycles produce better accuracies, without overfitting.
- The learning rate of the models affect accuracy more. Low learning rates are preferred, and high learning rates achieve very low accuracies near 0%.
- Lower confidence thresholds seem to make the models perform significantly better. High confidence thresholds tend to significantly reduce the accuracy of the models
- All models show better accuracy with simpler network structures, using 4 convolution layers or less.

6.4 Conclusion Research Question 3

The **Single Note** model performs adequately on the validation and test sets in a computer environment and on real time data when embedded on a device with an accuracy of around **78%** on the former and **70%** on the latter. The **Interval** recognition model performs significantly worse in both environments, with accuracies of circa **30%** and **17%**, respectively. Both the **Interval Root** note and **Chord** recognition models perform decently on the computer, but far from desirable on the IoT device. The former shows accuracies of around **60%** and **33%** respectively, while the latter shows accuracies of around **60%** and **47%** respectively.

6.5 Future Work

Further work stemming from this research should primarily consist of bigger tests with a custom audio processing block that puts more emphasis on recognized frequencies as well as tests with bigger models, more data, and the possibility for longer run times. Furthermore, it is essential for the research of this system that more experimentation is done to determine optimal parameters and network structures for the models and to further chart sacrifices that might need to be made as a result of limited capabilities of the target device and the viability for real-time recognition.

7 ACKNOWLEDGEMENTS

I would like to thank Yanqiu Huang for her excellent supervision and feedback throughout this research and Gertwillem Römer and Wietse Uittenbogaard for their assistance with data acquisition, even though their specific data samples were not used due to time constraints.

APPENDIX A: FIGURES AND TABLES FOR RELEVANT MUSIC THEORY

Table 2. Flat notes and their Enharmonic Sharp Equivalent Notes

Flat note	Enharmonic Sharp Equivalent
D♭	C#
E♭	D#
G♭	F#
A♭	G#
B♭	A#

Table 3. Intervals and distances between notes that comprise them.

Interval name	Example	Distance in number of notes (Semitones)
Minor 2nd	C – C#	1
Major 2nd	C – D	2
Minor 3rd	C – D#	3
Major 3rd	C – E	4
Perfect 4th	C – F	5
Tritone	C – F#	6
Perfect 5th	C – G	7
Minor 6th	C – G#	8
Major 6th	C – A	9
Minor 7th	C – A#	10
Major 7th	C – B	11
Octave	C – C	12

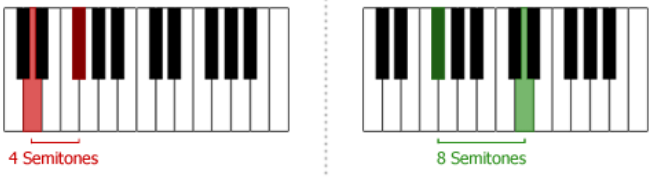

Figure 8. Schematic representation of an inversion of a Major 3rd to a Minor 6th interval through moving the lower note one octave up

REFERENCES

[1] Willis, G. (1998.). Introduction. In *Ultimate Ear Training For Guitar And Bass* (pp. 3–3). essay, Hal Leonard Corporation. Retrieved June 23, 2022, from https://kupdf.net/download/guitar-amp-bass-book-gary-willis-ultimate-ear-training-for-guitar-and-basspdf_5b1e16f4e2b6f5513ef77abc_pdf.

[2] Rizqyawan, M. I., & Hermawan, G. (2016). 2015 International Conference on Automation, Cognitive Science, Optics, Micro Electro-Mechanical System, and Information Technology (ICACOMIT). In *IEEE Xplore*. Bandung; IEEE. Retrieved June 23, 2022, from https://ieeexplore.ieee.org/document/7440200/references#references.

[3] Jensen, K., & Andersen, T. H. (2004). Computer Music Modeling and Retrieval . In *SpringerLink*. Montpellier. Retrieved June 29, 2022, from https://link.springer.com/chapter/10.1007/978-3-540-39900-1_2.

[4] Voinov, N. V., Ivanov, D. A., Leontieva, T. V., & Molodyakov, S. A. (2021). 2021 XXIV International Conference on Soft Computing and Measurements (SCM). In *IEEE Xplore*. St. Petersburg; IEEE. Retrieved June 23, 2022, from https://ieeexplore.ieee.org/document/9507134.

[5] Korzeniowski, F., & Widmer, G. (2016). 2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP). In *IEEE Xplore*. Vietri sul Mare; IEEE. Retrieved June 23, 2022, from https://ieeexplore.ieee.org/abstract/document/7738895.

[6] Sigtia, S., Benetos, E., & Dixon, S. (2016). An End-to-End Neural Network for Polyphonic Piano Music Transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *24*(5). https://doi.org/10.1109/TASLP.2016.2533858

[7] Black, T. R., & Donohue, K. D. (2002). Proceedings of the IEEE SoutheastCon 2000. 'Preparing for The New Millennium' (Cat. No.00CH37105). In *IEEE Xplore*. Nashville, TN; IEEE. Retrieved June 23, 2022, from https://ieeexplore.ieee.org/document/845433.

[8] Huang, T., & Yu, Y. (2018). 2018 International Conference on Audio, Language and Image Processing (ICALIP). In *IEEE Xplore*. Shanghai; IEEE. Retrieved June 23, 2022, from https://ieeexplore.ieee.org/abstract/document/8455558.

[9] Kuhn, W. B. (1990). A Real-Time Pitch Recognition Algorithm for Music Applications. *Computer Music Journal*, *14*(3), 60–71. https://doi.org/10.2307/3679960

[10] Kashino, K., & Murase, H. (2002). Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181). In *IEEE Xplore*. Seattle, WA; IEEE. Retrieved June 23, 2022, from https://ieeexplore.ieee.org/document/679655.

[11] Tak, R. N., Agrawal, D. M., & Patil, H. A. (2017). International Conference on Pattern Recognition and Machine Intelligence. In *SpringerLink*. Springer, Cham. Retrieved June 23, 2022, from https://link.springer.com/chapter/10.1007/978-3-319-69900-4_40.