Lappeenranta-Lahti University of Technology
School of Engineering Science
Degree Programme in Computational Engineering
Data Analytics

Antti Vilkman

Pitch Correction of Human Singing Voice Using Neural Networks

Bachelor's Thesis

Supervisor: Arto Kaarna, Associate Professor

Abstract

Lappeenranta-Lahti University of Technology
School of Engineering Science
Degree Programme in Computational Engineering
Data Analytics

Antti Vilkman

Pitch Correction of Human Singing Voice Using Neural Networks

Bachelor's Thesis

2020

24 pages and 10 figures.

Supervisor: Arto Kaarna, Associate Professor

Keywords: voice, pitch, neural network

Pitch correction of vocals is an integral part of music production, and while there are automatic methods for the process, they tend to produce a recognizable sound and their usage can be clearly heard in the final audio. To preserve the natural nuances and expressive elements of human singing, the pitch correction usually needs to be done manually, which is time consuming work. Finding an automatic solution that fixes the out-of-tune sections in the vocal without affecting the natural qualities of the voice using a simple convolutional neural networks was the aim of this thesis. A dataset consisting of recorded vocals and synthesized out-of-tune versions of these recordings was used for the training of the model. The out-of-tune versions were created by applying random pitch shifts to the original recordings. The used model achieved similar levels of accuracy for the training, validation and test data sets. While capable of making correct predictions for some of the input data, overall the accuracy of the model wasn't high enough to use for the task. Using more data and gathering a non-synthetic dataset could help to improve the performance while retaining the simple structure of the used neural network.

Tiivistelmä

Lappeenrannan-Lahden Teknillinen Yliopisto
School of Engineering Science
Laskennallisen tekniikan koulutusohjelma
Data-analytiikka

Antti Vilkman

Lauluäänen vireenkorjaus neuroverkoilla

Kandidaatintyö

2020

24 sivua ja 10 kuvaa.

Ohjaaja: Arto Kaarna, Tutkijaopettaja

Avainsanat: ääni, äänenkorkeus, neuroverkko

Laulun vireenkorjaus on olennainen osa nykypäivän musiikin tuotantoa, ja vaikka siihen on olemassa automaattisia ratkaisuja, niiden tuottama äänijälki on selkeästi tunnistettavissa lopputuloksesta. Laulun sisältämien luonnollisten nuanssien ja ilmaisullisten elementtien säilyttämiseksi virettä on usein korjattava manuaalisesti. Tämän työn tavoitteena oli kehittää automaattinen järjestelmä, joka korjaa epävireiset kohdat laulussa vaikuttamatta sen luonnollisiin ominaisuuksiin, käyttämällä konvoluutioneuroverkkoja. Opetusdatana käytettiin nauhoitettuja lauluja ja niistä syntettisesti luotuja epävireisiä versioita. Epävireset versiot luotiin tekemällä alkuperäisten nauhoitteiden vireisiin satunnaisia muutoksia. Käytetty malli saavutti samantasoiset tarkkuudet koulutus-, validaatio- ja testidatalle. Vaikka osa mallin syötteille antamista ennustuksista oli oikeanlaisia, kaikenkaikkiaan mallin tarkkuus ei ollut riittävän korkea tehtävään. Suuremman datamäärän ja ei-synteettisen datajoukon kerääminen voisi parantaa suorituskykyä säilyttäen käytetyn neuroverkon yksinkertaisen rakenteen.

Contents

Symboli- ja lyhenneluettelo

b	Bins per octave in constant-Q transform
f_0	Lowest frequency in constant-Q transform
N_k	length of sample window in constant-Q transform
Q	Ratio of frequency to resolution in constant-Q transform
CQT	Constant-Q transform
MSE	Mean-squared error
SR	Sample rate

1 Introduction

1.1 Background

Pitch is one of the fundamental attributes of singing voice, with humans often perceiving in-tune singing as pleasant while out-of-tune singing is considered unpleasant. Pitch depends on the frequency of the sound, but while frequency can be measured, pitch is the human perception of the frequency and therefore depends on the individual who observes the sound.

In music, instruments are generally tuned to certain standard frequencies and most music is based around the relations between these frequencies and rules that guide their usage to a result that is generally considered pleasant. In western music the standard set of frequencies is called the equal temperament scale, where each frequency is produced by multiplying the previous frequency by 1.0595. The amount of multiplication steps required to get from one frequency to another is called an interval, where a step between adjacent notes is called a semitone. The frequencies in the equal temperament scale are usually further ordered into smaller scales that are used to define the allowed frequencies within a song.

While the perception of pitch depends varies individually, it is fairly easy for the observer to determine if a singer is in tune relative to an accompaniment. Mathematically the fundamental frequencies of the singer's voice can be compared to the accompaniment to determine if the singer is in tune. If the singer is out of tune, their voice can be artificially manipulated to sound in tune, more commonly referred to as pitch correction.

Pitch correction of vocals is an integral part of music production today. An automatic solution for pitch correction was presented in the late 1990s in the form of Auto-Tune by Antares Audio Technologies [16]. Auto-Tune and various similar methods of automatic pitch-correction calculate the main frequency of the signal and simply shift this frequency to make them sound in tune with the desired scale [20]. Modern solutions, such as newer versions of Auto-Tune [16] or Melodyne [2] allow the user to manually edit the pitch of each note using a graphical interface. To preserve the natural expressive elements of the recording, such as vibrato, manual pitch correction is often necessary. Manually tuning vocal tracks is often time consuming work, especially in cases where a large amount of tracks need to processed.

A neural network solution to pitch correction was proposed in [20], where a Convolutional Recurrent neural network was trained on a synthetically created data to predict the amount of pitch shift needed to correct each note. A recurrent structure allowed the neural network

to take the temporal structure of music into account when making predictions. In addition to providing the out-of-tune vocal track as an input to the model, the backing track was also used as an input to take the musical context of the song into account when making the corrections to the pitch of the vocal track. While the recurrent structure achieved promising results,the training of recurrent models takes a large amount of resources.

1.2 Research objectives and delimitations

The objective of this thesis was to create an automatic solution to natural sounding pitch correction by using a strictly convolutional neural network structure to achieve faster training on limited computational resources than a recurrent structure would yield. For this purpose the training was done with fairly limited computational resources.

Ignoring the temporal structure also focuses the model only on the relationship of the frequencies in the vocal performance and the backing track. The goal of the model was to predict the amount of pitch shift that needed to be applied to a note to correct the pitch of the note in relation to the backing track.

A delimitation on the experiments was the fact that the neural network had to be trained on synthetically created data which means that the data may not represent the real world task accurately.

1.3 Structure of thesis

Chapter 2 describes the theory behind digital audio signal and neural networks. Chapter 3 details the the processing that is applied to the data and the structure of the proposed machine learning model. Chapter 4 describes the implementation of the data processing, the used environment and the implementation of the neural network as well as the experiments that were conducted and the results of these experiments. Chapter 5 evaluates the results and discusses the limitations and possible improvements.

2 Theory

2.1 Audio Signal

Sound is in essence vibration in a medium [3], that can be captured using a microphone. The microphone then translates the vibrations to a continuous voltage signal which is commonly known as analog signal. To process audio signal with a computer, it needs to be translated to a digital signal by sampling, as the continuous analog signal can have an infinite number of values at any time, while a computer has limited space to store values in.

The process of sampling means measuring the voltage of the analog signal at time instances as shown in Fig. 1. The time instances are usually chosen with a set time interval. Using a large enough sample rate per second, the audio signal can be stored as a series of discrete values while still being able to reconstruct it as an analog signal retaining the sound of the original signal. With the digital signal having a discrete values at discrete time points, it can now be processed by a computer.

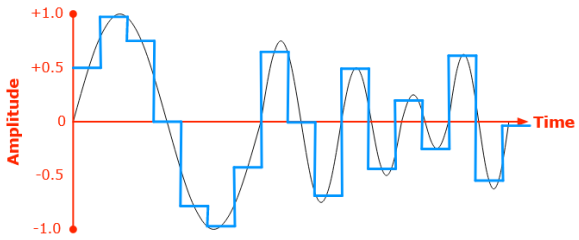

Figure 1. An illustration of a continous audio signal (the smooth wave) and a sampled version of it [4].

2.2 Artificial Neural Networks

Artificial neural networks are a class of machine learning systems that are based on the operations of the human brain in performing a given task and are capable of improving their performance by learning over time [6]. Artificial neural networks can be used for example in tasks involving classification and regression to create an algorithm that accurately models some observed phenomenom.

Neural networks consist of processing units called neurons such as the one shown in Fig. 2. A neuron takes input signals that have their own weights. The weighted input signals are

then summed in the neuron and the activation function along with the bias determines the output of the neuron based on the value produced.

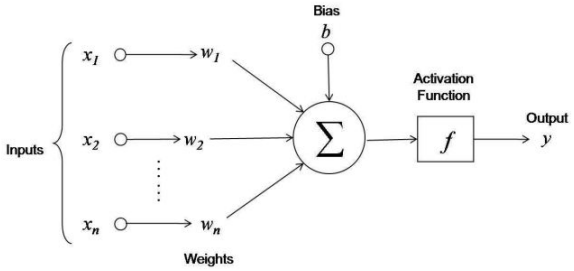

Figure 2. A model of a neuron [15].

A neural network is created by organizing neurons into layers and creating a network-like structure such as the one shown in Fig. 3. The architecture of the neural network varies depending on the task at hand. The input layer receives the raw input data, which is then fed into the hidden layers producing the output that is wanted for the chosen task. However, simply creating the network doesn't mean it will perform in the task it is given, as it needs to be trained.

In supervised learning, a neural network is trained by using data that contains both the inputs and the desired output corresponding to each input. By feeding this training data through the network and employing a learning algorithm that adjusts the weights in the neurons, the network can be trained to give the correct output even for inputs that it hasn't processed before.

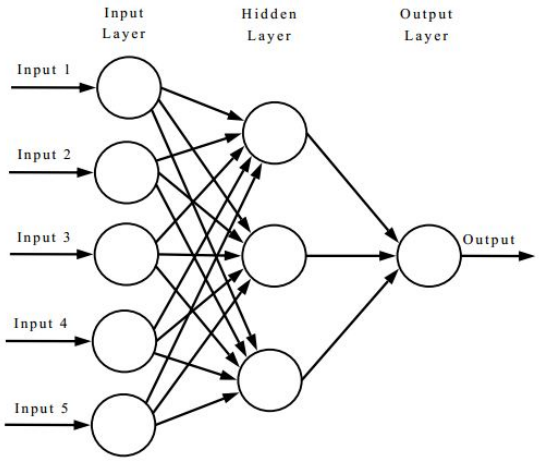

Figure 3. A basic artificial neural network [15].

3 Pitch Correction with a Neural Network

3.1 Pitch modification

In order to make pitch corrections that the model gives an algorithm needs to be used. A simple method of pitch modification from the Python LibROSA library [11] was used for this model. The pitch shift function from LibROSA is based on resampling of the audio signal. In order to shift the pitch of the audio signal, its sample rate is changed to speed up the signal or slow it down [13]. This has the side-effect of changing the length of the signal, and in order to preserve the original speed and length, the signal is stretched in time and then converted back to the original sample rate to achieve the same length as the original signal had.

3.2 Input Data Processing

In order to feed the recorded vocal to the neural network for tuning, it is divided into notes to mimic the representation in Fig. 4 where the signal has been divided into individual notes for pitch editing. The splitting is done by using the silence between each note to find the note boundaries. Every note in the input is processed by the neural network and the model predicts the amount of pitch shift needed to correct the note. If a note is in tune the predicted correction would be close to 0. After the model has made the predictions for each note in the signal the corrections are applied to the out-of-tune vocal track.

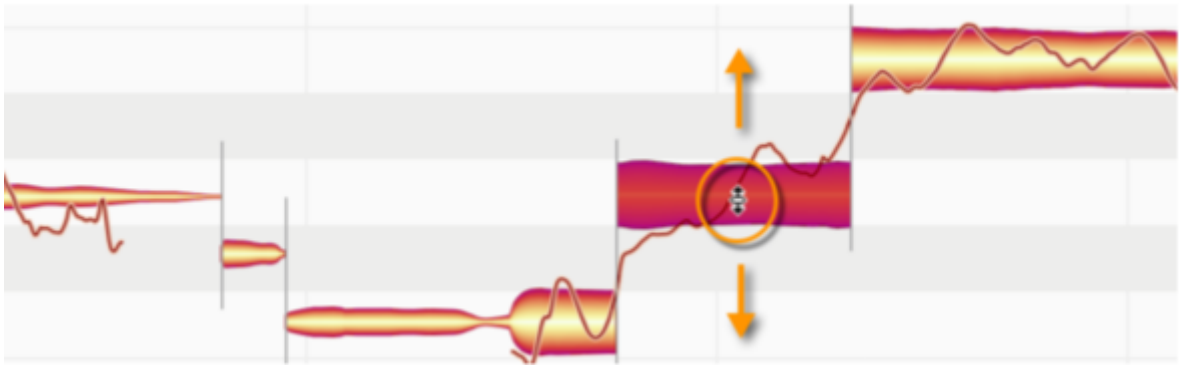

Figure 4. A graphic representation of notes and their pitches. The horizontal axis represents time and the vertical axis represents pitch [2].

As each sample in the audio signal is in essence only the voltage measured at that time

point [14], simply feeding the raw audio input doesn't allow the neural network to extract relevant frequency information from the signal, and therefore the inputs need to be converted into frequency domain. This is achieved by using the Constant-Q transform, or CQT, which is in essence a series of filters that can be mapped to musical notes by choosing the right parameters which makes it well suited for signal processing in music [1].

Calculating the Constant-Q transform requires a chosen minimum frequency f_0 and knowledge of the sample rate of the signal. Starting from f_0, the frequency that corresponds to the k-th CQ-bin can be calculated as

$$f_k = 2^{k/b} \cdot f_0 \tag{1}$$

where b is the chosen amount of CQ-bins per octave (12 steps in the equal temperament scale) and k is the total amount of filters to be computed. With this knowledge the CQT of a sequence x can be calculated as follows:

$$Q := (2^{1/b} - 1)^{-1} \tag{2}$$

is the ratio of frequency to resolution. Using Q the length of a window in samples can be calculated as

$$N_k := \frac{SR}{f_k} Q \tag{3}$$

where SR is the sample rate of the signal. Essentially at lower frequencies the window is larger.

The k-th component of the CQT is now

$$x^{cq}[k] := \frac{1}{N_k} \sum_{n<N_k} W[k,n] x[n] e^{-2\pi j Q n / N_k} \tag{4}$$

where $x[n]$ is the nth sample of digital audio signal and $W[k,n]$ is a window function.

With a high enough amount of constant-Q bins per musical note, the resolution of the CQT can be increased to represent very small deviations in frequency. As shown in Fig. 5, the shape of the pitch present in the vocal sound can be extracted using the CQT.

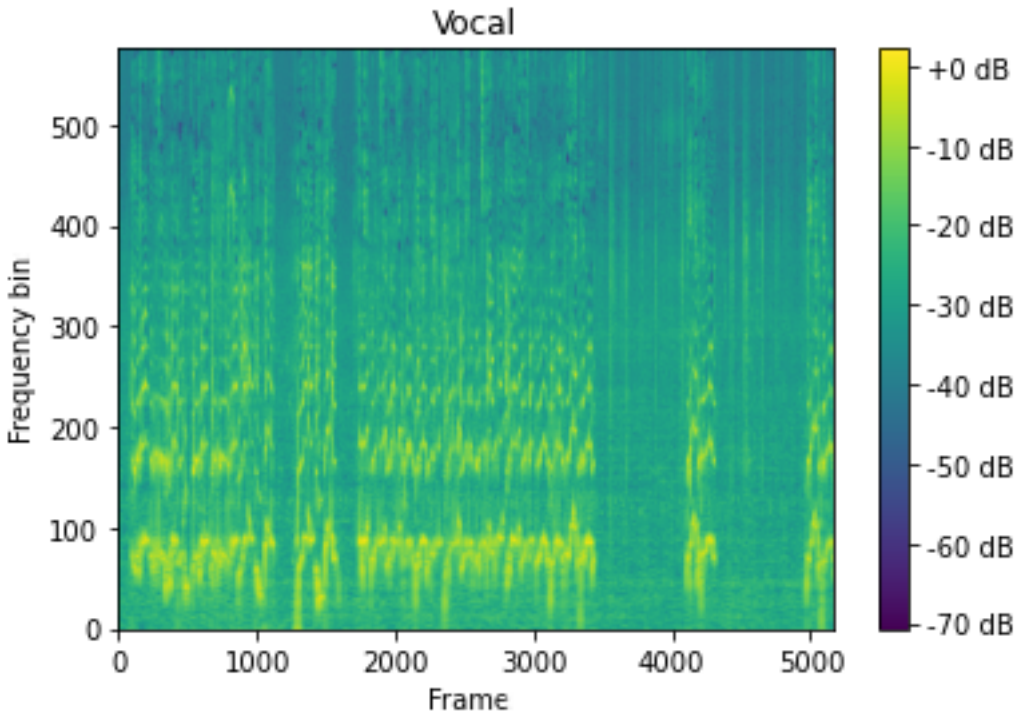

Figure 5. A Constant Q-transform of a 60-second long vocal track. Each frame in the horizontal axis corresponds to roughly 11ms. The vertical axis contains the frequency with the lowest frequency being 125 Hz. The outline of the melody and the overtones can be clearly seen in the bright yellow curve showing the louder frequencies.

The CQTs are split into notes by using the note boundaries computed from the raw audio. The notes are then padded to a fixed length to feed them to the neural network. Using the longest note in the set as the fixed length that notes are padded to would lead to having large amounts of zero padding in shorter notes wasting memory. Therefore a fairly short maximum length is used and longer notes are split into parts. An example of the final input data format is shown in Fig. 6 with the detuned note and the backing track.

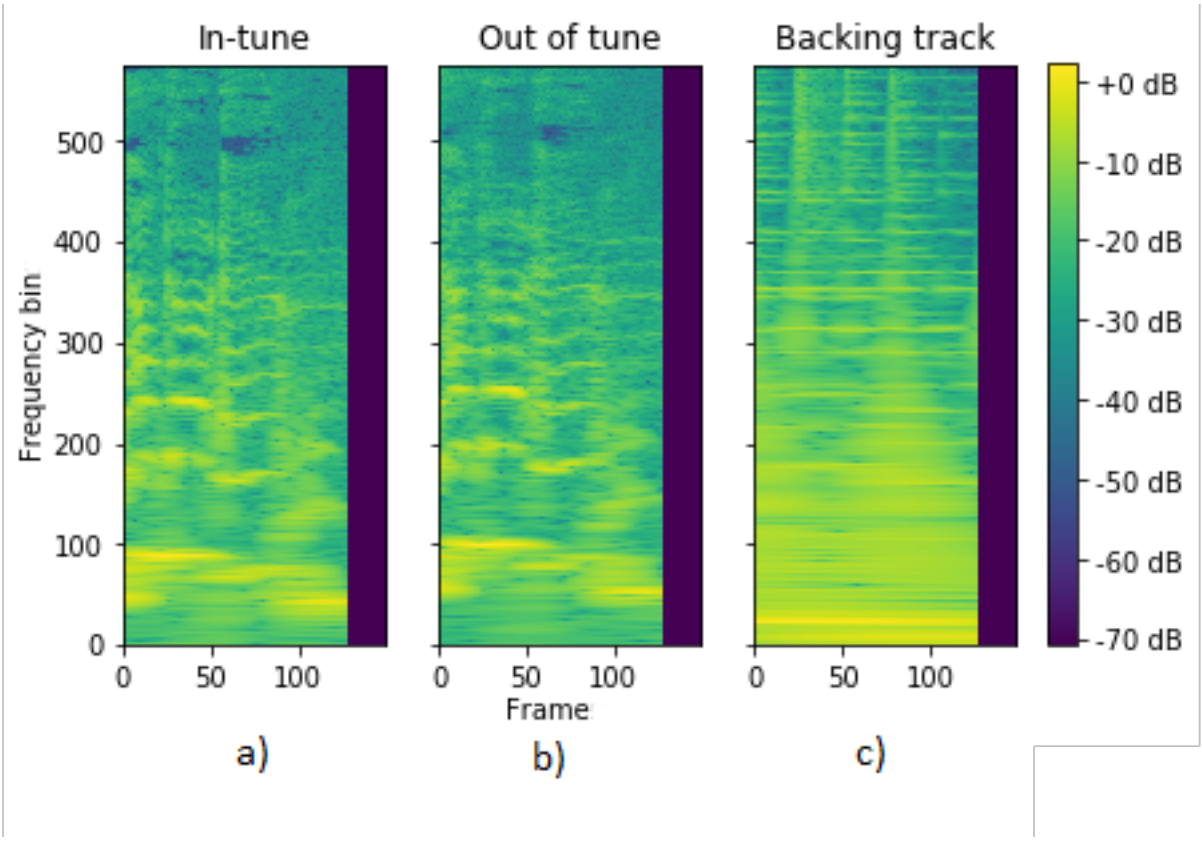

Figure 6. The input data format, with a) being the in tune vocal, b) the detuned version and c) the backing track. The applied shift can be seen most clearly around the 100th CQT bin in the out of tune note as the frequencies have moved slightly upwards. The blackness at the end is the zero-padding.

3.3 Convolutional Neural Network

For a task where extracting features and shapes such as the ones present in the CQTs a convolutional neural network is suitable. A convolutional neural network (CNN) is a type of neural network that is capable of extracting features such as shapes or patterns from the input it is given [10]. A typical CNN architecture such as AlexNet [9] consists of convolutional layers, pooling layers and fully connected layers. The convolutional layers contain filters that learn to recognize structures and shapes in the input data while pooling layers reduce the size of the output from the filters. After the convolutional filters have been applied their output is fed into a fully connected layer which produces the output of the model. Fig. 7 contains the concept of a CNN structure showing the convolutions and subsampling of the input as it progresses through the network.

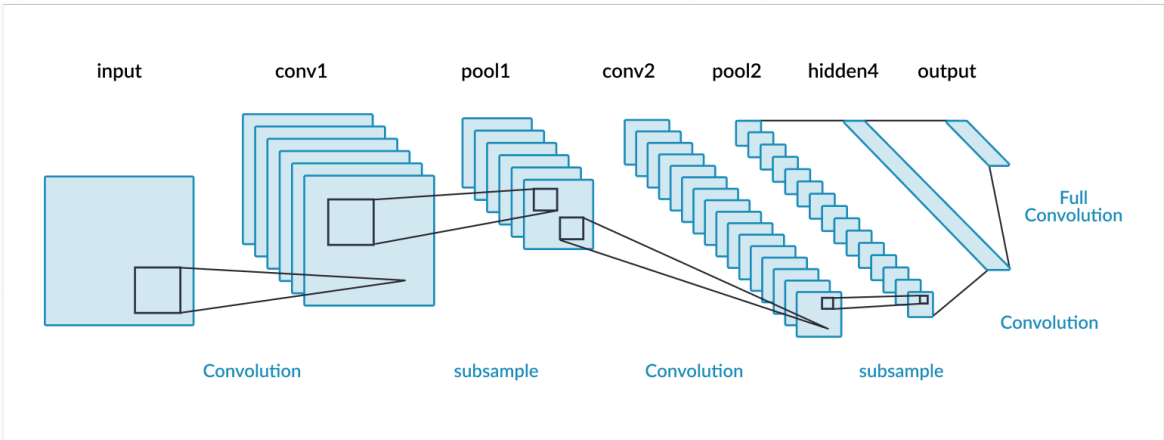

Figure 7. The concept of a CNN structure. From [12].

3.4 CNN in pitch correction

In the case of the pitch correction task, the neural network can be trained using raw vocal recording that contains out-of-tune notes, and a version of the recording that is considered to be well in tune. Using this data the model can be trained to apply the needed amount of pitch correction to these notes. To take the musical context of the song into account, frequency information from the backing track is also provided as an input to the model. The backing track is considered to be in tune.

The difference between the out-of-tune notes and the ones that are in tune is the output that the model needs to learn. The model applies the convolutional filters to the CQTs of the vocal recording and the backing track and aims to learn the right amount of pitch shift based on the frequencies present in the inputs.

4 Experiments

4.1 Data

The data used for training and evaluating the model was the 'Intonation' dataset created by Wager et al. (2019) for research in music information retrieval [19]. The dataset contains 4702 vocal tracks, as well as information extracted from the vocal tracks and the CQTs containing frequency information between 30 and 90 seconds from the backing tracks [5].

The performances in the dataset were considered to be in tune, and the out-of-tune signals were synthetically generated by splitting the signals into notes using a threshold of 20 dB and applying a random amount of pitch shift to every note. The pitch shifts were limited between -0.8 and 0.8 semitones to simulate skilled singers being able to avoid very large mistakes in their recordings.

After the synthesized out-of-tune vocal tracks were generated, they were normalized by their standard deviation as was done in [5], and the CQTs were then computed using LibROSA's CQT-function. The CQTs were computed using the same parameters that were used for the generation of the already computed backing track CQTs: f_0 of 125, 576 CQT bins with 96 bins per octave to achieve a resolution of 8 bins per semitone spanning 6 octaves, and a frame size of 256 samples which corresponds to roughly 11ms using the default sample rate of 22050 Hz in LibROSA's loading function.

The CQTs were then split according to the note boundaries and zero-padded to a maximum length of 150 CQT frames with longer notes being split into several. In the end the dataset used for training consisted of the CQT of each note and the corresponding CQT from the backing track such as the ones shown in Fig. 6, as well as the inverse of the shift that had been applied to the corresponding note.

4.2 Neural Network Structure

The model used for these experiments consisted of the vocal and the backing track input layers, reshaping layers for both inputs, a concatenation layer to combine both inputs to a two-channeled input and multiple convolutional layers gradually decreasing in kernel size. The goal of the structure was to start with extracting large features and gradually move onto extracting smaller features with a larger amount of filters. Batch normalization [8] was used after each convolutional layer to increase the stability of the network by normalizing the

activations of the layers, and max pooling was applied to reduce computational load.

After applying the convolutional filters the output was flattened to a single dimension and a single fully connected layer was used for the final prediction output. The diagram of the model structure is shown in Fig. 8. Convolutional layers have kernel sizes of 11, 7, 5 and 3 respectively. The amounts of filters in the layers are 32, 64, 128 and 128 respectively. Each pooling layer has a pool size of 2. The final output is floating point number predicting the amount of pitch shift that needs to be applied using the LibROSA pitch shifting function.

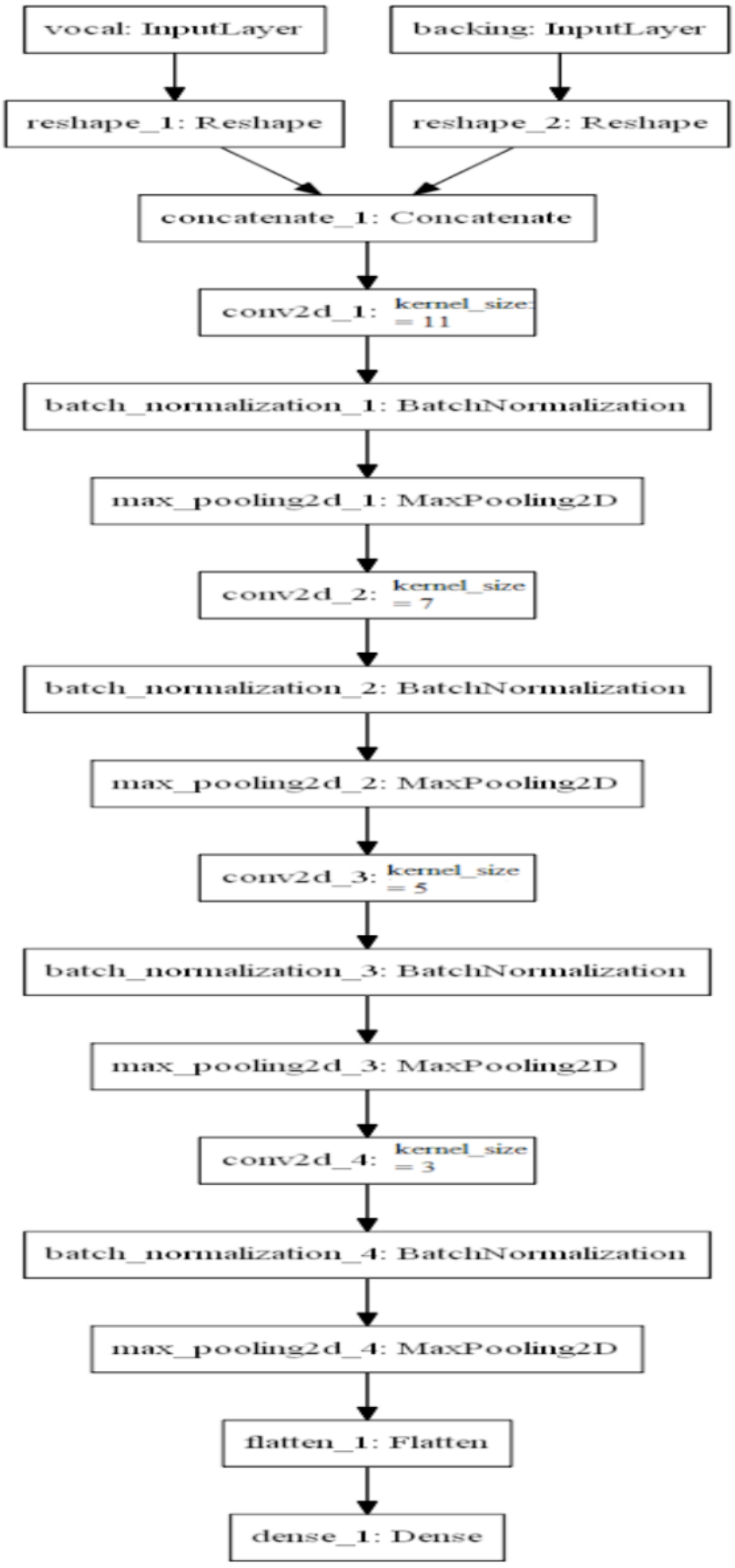

Figure 8. The proposed neural network structure.

4.3 Environment

The model was created in Anaconda3 Python environment [7], using Keras [17] with TensorFlow GPU backend. The LibROSA library [11] was used for feature extraction and manipulating the audio data. The training was done using a system with an Intel i5-4670k CPU, 16 GB of RAM and an NVIDIA GTX 1060 GPU with 3 GB of memory.

4.4 Results

The model was trained on a set of notes taken from the first 900 songs from the list of performances provided with the dataset. A validation set taken from the next 100 songs was used to evaluate the performance of the model during training. As the entire set of training data didn't fit to memory, the training was done by loading subsets consisting of 100 songs with each actual epoch consisting of running a single epoch on each subset.

For the training Adam optimizer was used with a learning rate of 0.0001 and mean-squared error (MSE) was used as the loss metric. The model was trained for 34 epochs on the entire training set and validation loss was measured at the end of each epoch. The training loss was taken from the last subset of each epoch. The used codes are publicly available at [18].

The losses during training are shown in Fig. 9. Initially both training loss and validation loss decrease but on later epochs validation loss starts to increase while training loss keeps decreasing which is a sign of the model overfitting to the training data and not properly learning to generalize. In the end the model achieved a training loss of 0.1901 while the validation loss is 0.2153. Additionally a test set of 50 songs was taken from the performances 3000 to 3050 on which the MSE was 0.1888.

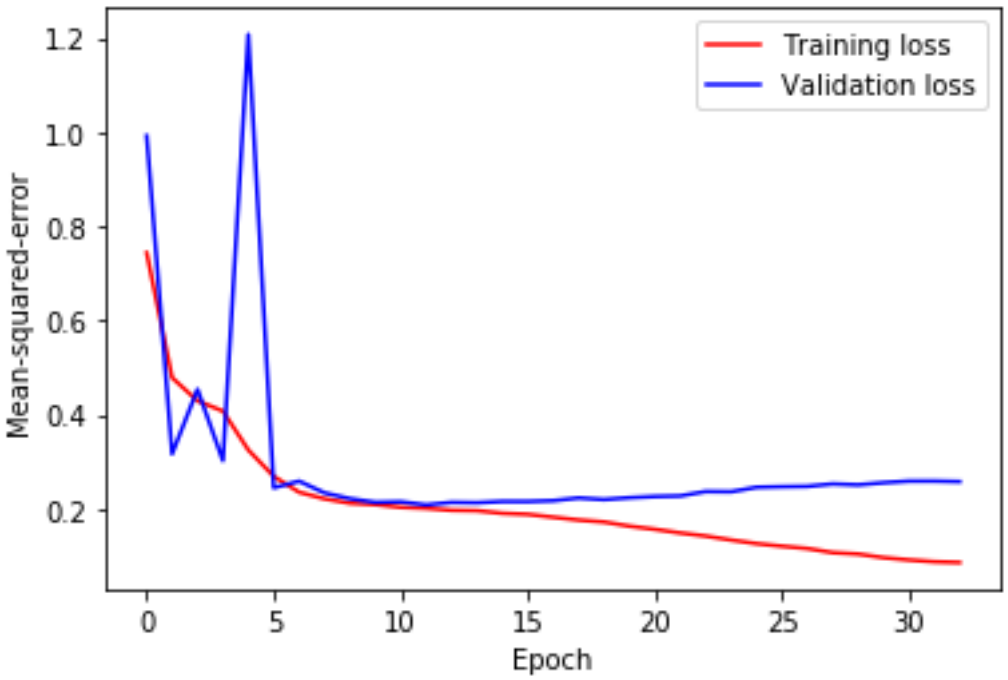

Figure 9. Results from the training of the neural network.

Additionally the first 100 predictions the model made on the test set were plotted alongside the target shifts to visualize the results. While some predictions are very close to the target, most are off by a fair margin and some shifts are predicted in the wrong direction, increasing the detuning.

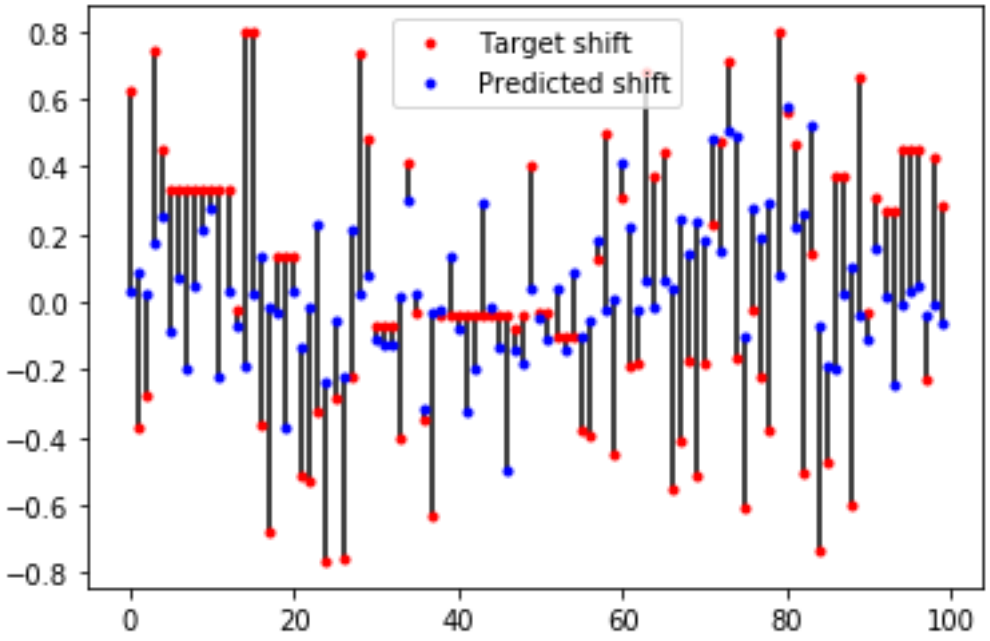

Figure 10. The comparison between the target shifts and the predictions made by the model.

In comparison, in [20] training loss of 0.049 and validation loss of 0.062 was achieved using 4561 songs for training and 49 for validation with seven differently pitch shifted versions of each song. The pitch shifting was done using a maximum of an entire semitone while for this thesis the pitch shifting was limited between -0.8 and 0.8 semitones making the performance of the model in [20] much better in relation as the training would start with a larger loss.

5 Discussion

The used data doesn't accurately describe the natural errors that may occur in human singing, as the out-of-tune sections were artificially generated. The used pitch shifting and note splitting methods were fairly simple and computationally efficient, which meant that their use could clearly be heard in the synthesized tracks. To avoid any artefacts produced by the synthesizing process affecting the performance of the model and to represent the task of applying pitch correction to vocals more accurately, a dataset consisting of raw vocal recordings and manually pitch corrected versions of these recordings would be required.

As the amount processing power and memory available for the training of the neural network was fairly limited, both the size of the neural network and the amount of data used had to be limited to an amount that could be reasonably processed using the available resources.

While the model achieved similar losses on the training, validation and test datasets, the predictions made by the model show that the accuracy isn't high enough to consistently correct the tuning of the input. With a different model and more training data however, successful pitch correction can be achieved as was shown in [20]. Their usage of a recurrent neural network structure allows the model to use each CQT-frame's position as a part of a sequence to better predict the shape of the melody.

6 Conclusion

The goal of this thesis was create an automatic solution for pitch correction using a fairly simple convolutional neural network. Due to limited computational resources a limited amount of data had be used for training. Due to the relatively small amount of data and its synthetic nature, the model failed perform well with new data presented to it. While the simple model wasn't capable of achieving the desired results, other research has shown that with a more complex structure, neural networks can achieve promising results in the realm of pitch correction. Gathering a non-synthetic data set to train the model on would also be an important step towards better performance.

References

[1] Judith C. Brown. “Calculation of a constant Q spectral transform”. In: *Journal Of The Acoustical Society Of America, 89(1), pp. 425-434* ().

[2] Celemony. *Melodyne 4 user manual*. URL: http://helpcenter.celemony.com/pdf/melodynestudio4_English.pdf (Accessed Feb. 20, 2020).

[3] Mads Christensen. *Introduction to Audio Processing*. Cham: Springer International Publishing, Jan. 2019. ISBN: 978-3-030-11780-1. DOI: 10.1007/978-3-030-11781-8.

[4] MDN contributors. *Digital Audio Concepts*. 2019. URL: https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Audio_concepts (Accessed Apr. 25, 2020).

[5] Perry Cook and Sanna Wager. *DAMP Intonation Dataset*. 2018. URL: http://CCRMA.stanford.edu/damp (Accessed Apr. 16, 2020).

[6] Simon Haykin. *Neural Networks and Learning Machines (3rd edition)*. Pearson, 2008. ISBN: 978-0-13-147139-9.

[7] Anaconda Inc. *Anaconda*. 2020. URL: https://www.anaconda.com/ (Accessed Apr. 30, 2020).

[8] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. JMLR.org, 2015, pp. 448–456.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Neural Information Processing Systems* 25 (Jan. 2012). DOI: 10.1145/3065386.

[10] Thomas Lidy and Alexander Schindler. “CQT-BASED CONVOLUTIONAL NEURAL NETWORKS FOR AUDIO SCENE CLASSIFICATION”. In: *Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*. Sept. 2016.

[11] Brian McFee, Colin Raffel, Dawen Liang, Daniel Ellis, Matt Mcvicar, Eric Battenberg, and Oriol Nieto. “librosa: Audio and Music Signal Analysis in Python”. In: Jan. 2015, pp. 18–24. DOI: 10.25080/Majora-7b98e3ed-003.

[12] missinglink.ai. *Convolutional Neural Network Architecture: Forging Pathways to the Future*. URL: https://missinglink.ai/guides/convolutional-neural-networks/convolutional-neural-network-architecture-forging-pathways-future/ (Accessed Apr. 30, 2020).

[13] Allam Mousa. “Voice Conversion using Pitch Shifting Algorithm by Time Stretching with PSOLA and Re-Sampling”. In: *Journal of Electrical Engineering* 61 (June 2011), p. 2011. DOI: 10.2478/v10187-010-0008-5.

[14] Panos Photinos. *Musical Sound, Instruments, and Equipment*. 2053-2571. Morgan Claypool Publishers, 2017. ISBN: 978-1-6817-4680-7. DOI: 10.1088/978-1-6817-4680-7.

[15] snives. *SimpleNeuralNetwork*. 2018. URL: https://github.com/snives/SimpleNeuralNetwork (Accessed Apr. 25, 2020).

[16] Antares Audio Technologies. *Auto-Tune Pro, User Guide*. URL: https://www.antarestech.com/mediafiles/documentation_records/Auto-Tune_Pro_Manual.pdf (Accessed Feb. 20, 2020).

[17] TensorFlow. *Keras*. 2020. URL: https://www.tensorflow.org/guide/keras (Accessed Apr. 30, 2020).

[18] Antti Vilkman. *Kandiautotuner*. 2020. URL: https://github.com/MieIi/Kandiautotuner (Accessed May 1, 2020).

[19] S. Wager, G. Tzanetakis, S. Sullivan, C. Wang, J. Shimmin, M. Kim, and P. Cook. “Intonation: A Dataset of Quality Vocal Performances Refined by Spectral Clustering on Pitch Congruence”. In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, pp. 476–480.

[20] S. Wager, G. Tzanetakis, C. Wang, and M. Kim. “Deep Autotuner: A Pitch Correcting Network for Singing Performances”. In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2020, pp. 246–250.