Understanding Music Representation in Neural Networks for Key Identification

Anna Greer

Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh
2019

Abstract

In order to outperform traditional key identification algorithms based on theory, we create neural network models which learn music representations for themselves and we analyse those representations.

Acknowledgements

I would like to thank my supervisor, Mark Steedman, for being so reliable and helpful throughout the writing of this dissertation. I am grateful to him for writing the paper that inspired this work and allowing me to be creative in my approach.

I would also like to thank Cameron McSorley for discussing every aspect of this project with me despite having no interest in music. His input has been invaluable and helped me understand what problem I was trying to solve.

I am also grateful for the help of Milo Stanojevic, who always met my neural network questions with a quick and thorough response. The Informatics Teaching Support at the University of Edinburgh were also essential in helping me fix technical issues related to this dissertation, for which I wish to thank them.

Finally, I would like to thank my friends and family for their unwavering support.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Anna Greer*)

Table of Contents

Chapter 1

Introduction

The key of a musical piece reveals the harmonic centre and expected patterns in harmonic rhythm of the piece, Korzeniowski & Widmer (2017b). Many tasks call for information on the content of a musical track, be it mood identification, chord recognition or perception research, Campbell (2011).

However, the key of a track is often ambiguous and key identification methods define the key in different ways. Classic key identification algorithms each define the key justified with music theory accumulated over centuries.

This dissertation aims to achieve two goals. The first is to create a key identification method using neural networks and little to no knowledge of music theory which outperforms music theory-based key identification algorithms, focussing on the Longuet-Higgins Steedman algorithm and the Temperley and Krumhansl key profile methods, (Longuet-Higgins & Steedman (1987), Temperley (1999), Krumhansl (2001)). The second is to analyse the layers of the model to interpret the information captured in the representations of music learned by the model.

We trained four dierent neural networks: a CNN and an LSTM each trained with two differing input formats. The first set of inputs consists of tracks with notes given as a pitch ranging from 0 to 11, beginning with 0 for C, which the network then learns a 2-dimensional embedding from (referred to hereafter as 'non-harmonic input'). The second encodes human knowledge in the form of the 2-dimensional harmonic network, which relates notes to each other in terms of major third and perfect fifth (referred to hereafter as 'harmonic input').

We use t-SNE to map the model predictions to a 2-dimensional space. This will allow us to see how all four kinds of neural networks see the key space. Which keys do they find to be related and so map close together? If we map the target keys to the points in this space, do they coincide with the model's vision? This also provides a visual aid in determining which mistakes the models are more likely to make.

Next, given that half of the models learned a 2-dimensional embedding of the input notes, we investigate these embeddings and compare them to the harmonic network. In particular, we are interested in the relationship between notes embedded close together and whether the CNN and LSTM learn different embeddings.

After seeing a general overview of what the networks learn, we compare the track-by-track performance of a modified LHS algorithm, harmonic CNN and harmonic LSTM on the forty-eight fugues of Bach's Well-Tempered Clavier. For each track, we aim to explain the mistake each algorithm has made, if any. We consider heat maps of the two neural networks' output probability vectors for each track, showing how confident each decision was and which other keys that were considered.

Finally, we take a closer look at the activations of the first layer of each neural network. Noting the patterns in per-note activations for each track can explain what the neural network pays attention to and why it may have failed.

We hope to further our understanding of music by letting the network show what it has learnt without needing to be taught, and display mistakes that could be made if not for the additional information provided by the theoretical underpinning of music.

Chapter 2

Background

2.1 Musical definitions

We define musical concepts mentioned in this paper. In particular, we explain the musical notation used and define words relating to key relationships. There are 24 possible keys in this project, corresponding to the twelve different pitches and their minor or major mode. The key scales of each possible key can be found in Appendix.

2.1.1 Note denomination

The data used in this project represents notes as pitches from 0 to 11. A single pitch can have a number of different names depending on the context. To simplify naming, pitches will only have one name in this project. These names and alternatives can be found in Table 2.1.

Pitch	0	1	2	3	4	5	6	7	8	9	10	11
Project notation	C	C#	D	E-	E	F	F#	G	A-	A	B-	B
Other notations	B#	D-	D	D#	F-	E#	G-	G	G#	A	A#	C-

Table 2.1: Denomination of musical notes.

2.1.2 Key relationships

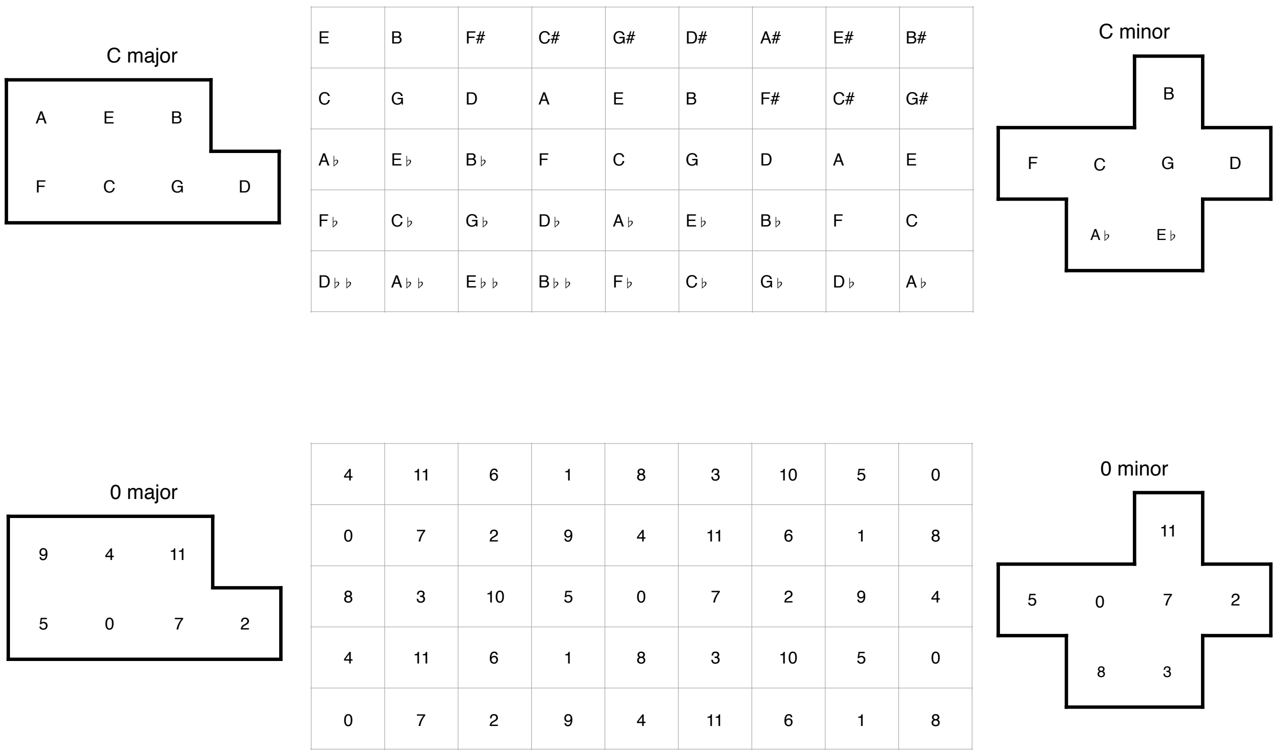
Definitions to musical vocabulary used to describe the relationship between keys:

- key scale: list of notes that make up a key

- tonic: the most frequent note in the track around which the melody is built, it is the first note listed in a key scale
- dominant: note a perfect fifth above a given note, it is the fifth note listed in a key scale
- semitone: the smallest interval in Western music, it is the distance between two notes in the twelve pitch scale
- perfect fifth: note seven semitones above a given note
- major third: note four semitones above a given note
- subdominant: the fourth note listed in a key scale, it is the same distance below the tonic as the dominant is above the tonic
- mode: refers to the distances between notes in a key scale (usually major or minor but not always)
- parallel key: key with same tonic, but different mode
- relative key: key of opposite mode with the same notes in the key scale
- dominant key: key whose tonic is the dominant of a given key
- subdominant key: key whose tonic is the subdominant of a given key

2.1.3 Harmonic network

Longuet-Higgins & Steedman (1987) describes a 2-dimensional lattice able to represent the harmonic structure of a piece of music. This grid structure, inspired by the Tonnetz in Euler (1739), is referred to as the harmonic network by Chew (2002). Given a note in the network, the note to its right is the major third. The note above is the perfect fifth. For a note C, these are G and E respectively (2.1). This network extends indefinitely, each note having an infinite amount of possible locations within it, see Figure 2.1.



E	B	F#	C#	G#	D#	A#	E#	B#
C	G	D	A	E	B	F#	C#	G#
A♭	E♭	B♭	F	C	G	D	A	E
F♭	C♭	G♭	D♭	A♭	E♭	B♭	F	C
D♭♭	A♭♭	E♭♭	B♭♭	F♭	C♭	G♭	D♭	A♭

4	11	6	1	8	3	10	5	0
0	7	2	9	4	11	6	1	8
8	3	10	5	0	7	2	9	4
4	11	6	1	8	3	10	5	0
0	7	2	9	4	11	6	1	8

Figure 2.1: Harmonic network expressed in notes and numbers, with the keys C major and C minor extracted, Longuet-Higgins & Steedman (1987).

2.2 Motivation for key identification

The key is a mid-level representation which captures information on the distribution of notes within a piece. If we know the harmonic centre of a piece, we know which patterns we can expect in the build-up and release of harmonic tension, Korzeniowski & Widmer (2017b). Many tasks call for information on the content of a musical track, be it music recommendation, chord recognition, audio matching or harmonic mixing, where a DJ finds tracks to mix by matching keys that complement each other, Campbell (2011). Automatically detecting the movements in tension in music also contributes to research in perception. In music, the most prominent debate is the cognitivist vs. emotivist view on emotion in music. Does the music express the emotion or does the music cause the listener to feel the emotion? Temperley & Tan (2013).

However, the key of a track is often ambiguous due to key modulations, when a key changes to another key within the piece, or because a selection of keys have very similar scales. Classic key identification algorithms each define the key in a different way justified with music theory accumulated over centuries. Using neural networks, we can free ourselves from preconceived notions of the key and let a machine learning model learn to represent the key for itself. We can then observe whether or not the representation of music it learns matches that of music theory.

2.3 Key identification algorithms

The following key detection algorithms are methodical, not data-greedy. Instead of relying on finding the correct hyperparameters so that a model can learn to predict the key itself, these algorithms are founded on human knowledge about music.

2.3.1 Longuet-Higgins Steedman

The Longuet-Higgins Steedman algorithm described in Longuet-Higgins & Steedman (1987) is a rule-based method. For each note in a track, we eliminate all keys that do not include this note. When only one track remains, that is the track of the key. If more than one key remains, call the tonic dominant rule. This rule states that if the first note of a track is the tonic of a key, that is the key of the piece. If not, then if the first note of the track is the dominant of a key, that is the key of the piece. If all keys are eliminated, all keys are reinstated and we continue our journey moving on to the next note in the track.

In the original paper, the algorithm includes a minor scale rule based on melodic convention of Bach's time. The minor scale rule denotes that if there is an ascending or descending scale in a minor key, the composer would use the major sixth in the ascending scale and the minor seventh in the descending. This means that notes not traditionally included in the minor key would be considered as part of the minor key. To use the example given in Longuet-Higgins & Steedman (1987), in C minor the ascending scale [G,A-,B] would become [G,A,B], while the descending scale [C,B,A-] becomes [C,B-,A-]. The algorithm correctly identifies the global key of all 48 fugues of the Well-Tempered Clavier.

2.3.2 Temperley and Krumhansl

The Krumhansl-Schmuckler algorithm uses the Pearson correlation values (def 2.3.1) between a distribution of the twelve pitches within a track and each of 24 key profiles created by calculating the correlation between key and note frequency on a training dataset. The key profiles were created using human expertise to annotate the key and by calculating the correlation between key and note frequency on a training dataset, (Krumhansl (2001)). The Krumhansl algorithm found the correct key on 44 of the 48 Well-Tempered Clavier fugues. This method is intuitive and shows a wider set of working cases than the LHS method built for the Well-Tempered Clavier. Temperley (1999) is a modified Krumhansl algorithm with adjusted key profiles and which did not depend on note frequency in the piece gives better results. This algorithm finds the correct key on 43 of the 48 fugues. Table 2.2 gives a comparison of the key profiles of these two methods. The Krumhansl algorithm has a strong tendency to identify the dominant key as the tonic, while the Temperley algorithm has a tends to misclassify minor keys as their relative major, but is well-balanced for major keys, (Sapp (2005)). In this context of key profiles, the LHS algorithm is a flat key-profile matching algorithm assuming all notes are equally as likely to appear, as opposed to Krumhansls weighted key-profile algorithm.

Definition 2.3.1. Pearson correlation

$$R(x,y) = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2(y_i - \overline{y})^2}}$$

x is a histogram of pitches in a musical piece. y is a list of twelve weights unique to each key corresponding to the scale of twelve pitches. $\overline{x}$ and $\overline{y}$ are the mean for each input (Krumhansl).

2.4 Key identification using machine learning

2.4.1 LSTM

LSTM networks, Hochreiter & Schmidhuber (1997), are a variant of Recurrent Neural Network (RNN) currently very popular for modelling sequential data, see Vinyals et al. (2014), Sutskever et al. (2014) or Graves (2013).
These networks are designed to make use of weights used in previous inputs when

Algorithm	Krumhansl		Temperley	
Tonic\Mode	Minor	Major	Minor	Major
C	6.33	6.35	0.712	0.748
C#	2.68	2.23	0.084	0.060
D	3.52	3.48	0.474	0.488
E-	5.38	2.33	0.618	0.082
E	2.60	4.38	0.049	0.670
F	3.53	4.09	0.460	0.460
F#	2.54	2.52	0.105	0.096
G	4.75	5.19	0.747	0.715
A-	3.98	2.39	0.404	0.104
A	2.69	3.66	0.067	0.366
B-	3.34	2.29	0.133	0.057
B	3.17	2.88	0.330	0.400

Table 2.2: Krumhansl and Temperley key profiles.

training on the current input. When used for predicting the next word in the sequence, such as in Zaremba et al. (2014), the network outputs a probability distribution of all possible words. We will adapt this problem to output probabilities for each of the 24 keys at each step.

Indeed, a language can be seen as a set of words used together in a typical order. Similarly, a musical key is a set of notes used to write a musical piece. Tracks of the same key tend to show the same patterns. This piece can contain notes from other keys, which are accidentals, just as a language occasionally incorporates words from other languages. When we identify a language in a text, we are a classifying this text. Thus, it makes sense to use an LSTM implementation used for text classification for the task of key identification.

2.4.2 CNN

Although the LSTM specialises in learning from sequential data, the CNN excels in the analysis and recognition of patterns invariant to translation. CNNs pool weights between nodes of the network to make use of the fact that similar inputs will often warrant similar outputs, making a CNN computationally more efficient and robust than

a vanilla neural network. Problems this technique is suited for include image recognition, annotation and regression as seen in Redmon et al. (2016), Long et al. (2015) and Zhang et al. (2014). CNNs have also shown promise in classification, from text to bird calls, (Sapp (2005), Sapp (2005), Sprengel et al. (2016)).
Music Information Retrieval Evaluation eXchange (MIREX) submissions for key recognition, Schreiber (2017) and Korzeniowski & Widmer (2017a), show that a CNN can be trained automatically for key identification without the need for any knowledge of music theory. Shuvaev et al. (2017) applies a CNN successfully to classical composer classification. To identify a composer from a same genre, the model must be able to recognise features which distinguish a composer from the others. and can then be leveraged for other tasks. The CNN is thus suited for the task of key identification, which is a classification task.

2.5 Neural network interpretation

Not only do we want to correctly identify the key of a musical piece, we also want to understand what features the neural networks have learnt. Music relies on a syntactic structure made of harmonic progressions and rhythm patterns in melodies that respect a grammar much like language, (Granroth-Wilding & Steedman (2014), Baroni (1984), Baroni et al. (1983)).
The field of distributional semantics, which aims to use data analysis to characterise the semantic relationship between words, has been very successful. Word embeddings such as *word2vec* or those listed in Turian et al. (2010) are now widespread, representing words as ready vectors capturing relationships that have semantic meaning. The note embedding learnt by the non-harmonic model should also capture semantic meaning in the language of music. The harmonic network is a precomputed note embedding encoding information about the relationship between notes.
Predictions of a classification model can also be used as an embedding of the target from which we can extract meaning. The models output a 24-dimensional softmax probability distribution from which we take the maximum probability as the key. We use t-SNE dimensionality reduction technique to map these 24-dimensional predictions to a 2-dimensional space, Maaten & Hinton (2008). Using this technique, two points close together in the original data will be close together in a set position in the embedding; the placement of two points far apart in the original data will be more approximate in the embedding.

Automatic feature extraction is a growing field of research, aiming to create neural networks which automatically create useful features from the data. The activations of a deep belief network have been used as a feature to create music genre recognition and auto-tagging models that perform better than networks using classic audio features such as timbral information of MFCCs (Hamel & Eck (2010)). Interpretive techniques are being developed to understand neural networks trained for machine translation and speech recognition,(Belinkov & Glass (2017), Belinkov et al. (2018), Dalvi et al. (2017)). Sprengel et al. (2016) relies on feature generation to create an end-to-end bird call classification system.
Intermediate layer interpretation is most commonly associated with CNNs applied to computer vision problems while RNNs are seen as uninterpretable black boxes, van der Westhuizen & Lasenby (2017). For both networks, intermediate layer activations can still provide interesting insights into the key features the neural network takes into account when applied to natural language processing or language-like problems such as music, (Simonyan et al. (2013), Karpathy et al. (2015), Li et al. (2015), Strobelt et al. (2016)).
Using the ideas mentioned, we can acquire a general overview of what the networks learn and understand how it classifies the key.

Chapter 3

Methodology and Implementation

3.1 Framework

The neural networks trained as part of this project were coded in Keras and run on a K80 Tesla GPU using the Google Cloud AI Platform.

3.2 Datasets

The datasets used in this project are obtained from parsing MIDI files. MIDI is a format that does not contain sound, but does contain instructions telling the computer how to play the tune, making it straightforward to extract note and key information.
The dataset used for training is made up of three datasets. The largest dataset is the Lakh MIDI file dataset by Colin Raffel (2016), of which 45,129 tracks of various contemporary genres have been matched to entries in the Million Song Dataset, Bertin-Mahieux et al. (2011). The genre distribution contains approximately 30% rock, 25% pop, 10% country, 5% r&b, 5% classical, 5% jazz and less than 5% dance. Another large dataset we will use is the music21 dataset Cuthbert, M and Ariza, C and Hogue, B and Oberholtzer, J (2017). This dataset comes with the music21 Python library, which the Massachusetts Institute of Technology created to allow for easy MIDI file parsing. The dataset contains thousands of annotated musical pieces from a selection of six folk and twenty-one classical corpora.
Given its popularity as a test set for key identification, we will also test our algorithm on MIDI versions of the 48 fugues of Bach's Well-Tempered Clavier.

3.3 Preprocessing

3.3.1 Input

3.3.1.1 Non-harmonic

The MIDI files were parsed to convert each track to lists of pitches numbered from 0 to 11 as described in Table 2.1.

3.3.1.2 Harmonic

The harmonic embedding of the input was created by mapping the lists of pitches of the non-harmonic input to the harmonic network. The algorithm is initiated by choosing one location of C as a starting point. Then, each note in the track is successively placed at a possible location closest to the notes before it using the Manhattan city-block distance. For two points P and M in a 2-D space with coordinates (x,y), it is defined as

$$d(M,P) = |M_x - P_x| + |M_y - P_y|.$$

The resulting list of 2-dimensional coordinates is used as harmonic input to the models in this project.

3.3.2 Output

The output used to represent the target keys changes the task the neural network is optimised for. Two different kinds of output are tested to observe how the performance is affected.

3.3.2.1 Categorical classification

The categorical classification output represents each target key as a one-hot 24-dimensional vector, one dimension for each key. That is, each target vector is made up of zeroes, except for the target key's position which is set to 1. Categorical classification means each key is treated as separate from the others, so we do not encode any information on how the keys are related. The last layer of a categorical classification model is a softmax, giving a probability distribution over the 24 keys. We take the key with highest probability as the predicted key.

3.3.2.2 Binary classification

The binary classification output represents each target key as a 13-dimensional vector, one dimension for each pitch and one dimension for the mode. Each feature is 0 or 1 depending on whether that feature is present in the target key scale. If C is in the key scale, then the first feature of the output will be set to 1. If the key is major, the last feature of the vector is set to 1 and set to 0 if it is minor. The model matching this output uses a sigmoid output layer, providing each feature with a separate probability of being presented or not. The binary classification output is meant to encode the fact that different keys have overlapping key scales, so if the model predicts a key close to the target, it will not be penalised as much as a completely different key. To get one predicted key from this output, we use the total probability rule to get a probability for each possible key. We multiply the probabilities that each note that is in the key scale is present by the probability that notes in the key scale are not present as well as the probability of the correct mode being predicted. We assume each feature is independent from the others. For a key k and pitches n between 0 and 11, the probability $P(k)$ of the key being k is:

$$\begin{aligned} P(k) \propto I(mode\ is\ minor) * (1 - P(mode\ is\ major)) * I(mode\ is\ major) * P(mode\ is\ major) \\ * \prod_{n=0}^{11} I(n\ in\ k) * P(n\ is\ present) + I(n\ not\ in\ k) * P(n\ is\ absent) \end{aligned} \tag{3.1}$$

3.4 Baseline Methods

3.4.1 Modified Longuet-Higgins Steedman

As a melodic convention of a certain time, the minor scale rule is not applicable to the wide range of music in the dataset and so is not included in the modified Longuet-Higgins Steedman (mLHS) algorithm. This does mean that the performance of the mLHS will suffer on minor keys of the Well-Tempered Clavier. See Appendix A.1 for the pseudo-code of the modified Longuet-Higgins Steedman algorithm.

3.4.2 Temperley and Krumhansl

The Temperley and Krumhansl algorithms are implemented using the Temperley-Kostka-Payne and Krumhansl-Schmuckler functions included in the music21 package. Implementation of weightings for key determination algorithm.

3.5 Models

3.5.1 Hyperparameter tuning

We tune the hyperparameters for each model with the scikit-optimise implementation of Bayesian optimisation. This method uses Gaussian processes to create a probabilistic model of the objective it wants to maximise, which in this case is accuracy. This optimisation method is more efficient than other hyperparameter search method such as grid search and random search (Bergstra et al. (2011), Shahriari et al. (2015), Dewancker et al. (2015)), for models that are expensive to evaluate, such as the ones of interest to this project.
All models use relu activation and an Adam optimiser. Categorical classification uses softmax activation in the output layer and the categorical crossentropy loss function. Binary classification uses sigmoid activation in the output layer and the binary crossentropy loss function.
We truncate the tracks to a maximum of 200 notes to increase efficiency as well as encourage key detection early in the track.

3.5.2 LSTM

The LSTM models are inspired from code written for text classification Brownlee (2016).

3.5.2.1 Hyper-parameters

The LSTM hyper-parameters including in tuning:

- number of epochs (50 to 600),
- batch size (32 to 128),
- number of LSTM layers (1 to 12),

- number of LSTM cells (32 to 256),
- learning rate of the optimiser (1e-04 to 1e-06),
- truncated length of each track (20,50,100,200)

3.5.2.2 Architecture

LSTM	Layer
If non-harmonic	Embedding layer (2-dimensional)
Repeated as specified in parameters	LSTM(return_sequences = True, activation = relu)
	LSTM(return_sequences = False, activation = relu)
	Dense(num_classes)

Table 3.1: LSTM architecture throughout this paper.

3.5.3 CNN

3.5.3.1 Hyper-parameters

The CNN hyper-parameters including in tuning:

- number of epochs (50 to 600),
- batch size (32 to 128),
- number of CNN layers (1 to 12),
- number of filters (32 to 256),
- number of kernels (5 to 8),
- learning rate of the optimiser (1e-04 to 1e-06),
- truncated length of each track (20,50,100,200)

3.5.3.2 Architecture

CNN	Layer
If non-harmonic	Embedding layer (2-dimensional)
Repeated as specified in parameters	Conv1D (activation = 'relu')
	GlobalAveragePooling1D
	Dropout(0.5)
	Dense(10, activation = relu)
	Dense(num_classes)

Table 3.2: CNN architecture throughout this paper.

3.6 Evaluation

Three evaluation measures are used to measure the performance of each model.

3.6.1 Accuracy

The first is accuracy, simply defined as the proportion of musical pieces for which the key was correctly identified. It is a simple but harsh metric for key identification, as many keys are similar or even the same in pitch membership. It is difficult for most humans to correctly identify the key of all music.

3.6.2 MIREX Score

The second is the error measure used in the MIREX key identification task, see Mardirossian et al. (2005). This measure relies on a point system to reward the proposed key based on its proximity to the target key:

This point system is more forgiving than accuracy as it rewards almost getting the key right.

3.6.3 Speed

In the context of this project, speed is the note at which the model identifies the key. A model which identifies a key earlier in a piece will present an advantage over its

Table 3.3: MIREX key identification point system.

Relation to correct key	Points
Same	1
Perfect fifth	0.5
Relative major/minor	0.3
Parallel major/minor	0.2

competitors for practical reasons.

In the LHS model, it is easy to report the speed, as it is going through a process of elimination, observing one note at a time to determine the key.

For the other models, it is not so straightforward. The Temperley and Krumhansl methods look at the distribution of notes in a whole track, comparing it to template distributions corresponding to each key. Similarly, a neural network is given a sequence of fixed length and then makes a prediction. These predictions are offered as probability distributions over the possible classes. This means that at any time the model can give you it's most likely guess, the key with the highest probability.

One way to investigate how quickly the method finds the correct key would be to feed the notes of the tracks one at a time. That is, first feed the model a track of length 1, then of length 2 and so on, incrementally providing a longer track. The model can at each stage offer a measure of confidence. We could decide to pass it a threshold and say when you find a probability over this threshold, declare this as the key.

That being said, it is not obvious what practical benefit there is to measuring how many notes it takes to make a confident prediction. This is especially the case for our models. Achieving at most an accuracy of 50%, a confident key would still probably be wrong.

Chapter 4

Results

4.1 Baseline Results

Baseline	Accuracy	MIREX
Modified LHS	0.28	0.33
Temperley	0.42	0.45
Krumhansl	0.40	0.43
Random	0.06	0.07

Table 4.1: Results of baseline methods on the training set.

Baseline	Accuracy	MIREX
Modified LHS	0.29	0.33
Temperley	0.42	0.45
Krumhansl	0.41	0.43
Random	0.06	0.07

Table 4.2: Results of baseline methods on the validation set.

As we can see in tables 4.1 and 4.2, the methods based on note frequency perform best, with the Temperley method performing best overall with an accuracy of 42% and MIREX score of 45% on both the training and validation sets. According to these results, it is possible that methods which incorporate pre-computed note frequencies in the definition of a key perform better than methods which assume all notes of a key can appear with equal frequency. The consistent scores over both datasets show

these methods are either robust to variance in the data or that the datasets contain the same variety of music. All of the baseline methods significantly outperformed random guessing.
On average, the modified Longuet-Higgins Steedman method found the key by the 23rd note in a track.

4.2 LSTM

4.2.1 Learned Embedding

Model	1	2	3
Epochs	200	200	200
Batch Size	64	64	64
Layers	5	2	1
Cells	128	128	128
Learning Rate	1e-04	1e-04	1e-04
Track Length	50	50	50

Table 4.3: Hyper-parameters of 3 learned embedding LSTM models with categorical classification targets.

Model	1	2	3
Accuracy	0.55	0.50	0.49
MIREX	0.59	0.58	0.56

Table 4.4: Results of learned embedding LSTM on the training set for categorical classification targets.

Model	1	2	3
Accuracy	0.45	0.43	0.42
MIREX	0.52	0.51	0.51

Table 4.5: Results of learned embedding LSTM on the validation set for categorical classification targets.

Model	1	2	3
Epochs	200	200	200
Batch Size	64	64	32
Layers	5	1	1
Cells	128	128	128
Learning Rate	1e-04	1e-04	1e-04
Track Length	50	50	50

Table 4.6: Hyper-parameters of 10 learned embedding LSTM models with binary classification targets.

Model	1	2	3
Accuracy	0.45	0.45	0.43
MIREX	0.54	0.54	0.52

Table 4.7: Results of learned embedding LSTM on the training set for binary classification targets.

Model	1	2	3
Accuracy	0.42	0.41	0.40
MIREX	0.51	0.50	0.49

Table 4.8: Results of learned embedding LSTM on the validation set for binary classification targets.

4.2.2 Harmonic Network Embedding

Model	1	2	3
Epochs	400	200	250
Batch Size	64	64	64
Layers	2	1	2
Cells	256	256	128
Learning Rate	1e-04	1e-04	1e-04
Track Length	50	50	50

Table 4.9: Hyper-parameters of 10 harmonic embedding LSTM models with categorical classification targets.

Model	1	2	3
Accuracy	0.67	0.47	0.40
MIREX	0.72	0.68	0.64

Table 4.10: Results of harmonic embedding LSTM on the training set for categorical classification targets.

Model	1	2	3
Accuracy	0.52	0.37	0.33
MIREX	0.58	0.57	0.55

Table 4.11: Results of harmonic embedding LSTM on the validation set for categorical classification targets.

Model	1	2	3
Epochs	200	275	164
Batch Size	64	64	64
Layers	5	8	9
Cells	128	32	64
Learning Rate	1e-04	1e-04	1e-04
Track Length	50	50	50

Table 4.12: Hyper-parameters of 3 harmonic embedding LSTM models with binary classification targets.

Model	1	2	3
Accuracy	0.15	0.15	0.14
MIREX	0.24	0.23	0.22

Table 4.13: Results of harmonic embedding LSTM on the training set for binary classification targets.

Model	1	2	3
Accuracy	0.17	0.16	0.15
MIREX	0.26	0.24	0.24

Table 4.14: Results of harmonic embedding LSTM on the validation set for binary classification targets.

4.3 CNN

4.3.1 Learned Embedding

Model	1	2	3
Epochs	400	400	400
Batch Size	64	64	64
Layers	5	12	8
Filters	128	128	128
Kernel Size	7	5	5
Learning Rate	1e-04	1e-04	1e-04
Track Length	200	50	50

Table 4.15: Hyper-parameters of 3 learned embedding CNN models with categorical classification targets.

Model	1	2	3
Accuracy	0.55	0.59	0.52
MIREX	0.63	0.67	0.60

Table 4.16: Results of learned embedding CNN on the training set for categorical classification targets.

Model	1	2	3
Accuracy	0.52	0.49	0.45
MIREX	0.59	0.57	0.54

Table 4.17: Results of learned embedding CNN on validation set with categorical classification targets.

Model	1	2	3
Epochs	400	400	400
Batch Size	64	64	64
Layers	10	10	10
Filters	128	128	128
Kernel Size	7	5	7
Learning Rate	1e-04	1e-04	1e-04
Track Length	200	200	50

Table 4.18: Hyper-parameters of 3 learned embedding CNN models with binary classification targets.

Model	1	2	3
Accuracy	0.60	0.56	0.55
MIREX	0.68	0.64	0.63

Table 4.19: Results of learned embedding CNN on training set with binary classification targets.

Model	1	2	3
Accuracy	0.50	0.48	0.46
MIREX	0.58	0.56	0.54

Table 4.20: Results of learned embedding CNN on validation set with binary classification targets.

4.3.2 Harmonic Network Embedding

Model	1	2	3
Epochs	200	400	200
Batch Size	64	64	64
Layers	5	5	5
Filters	128	128	128
Kernel Size	7	7	7
Learning Rate	1e-04	1e-04	1e-04
Track Length	200	100	100

Table 4.21: Hyper-parameters of 3 harmonic embedding CNN models with categorical classification targets.

Model	1	2	3
Accuracy	0.60	0.59	0.57
MIREX	0.70	0.70	0.67

Table 4.22: Results of harmonic embedding LSTM on the training set for categorical classification targets.

Model	1	2	3
Accuracy	0.55	0.53	0.52
MIREX	0.63	0.61	0.59

Table 4.23: Results of harmonic embedding CNN on validation set with categorical classification targets.

Model	1	2	3
Epochs	150	150	110
Batch Size	32	32	64
Layers	8	8	10
Filters	32	32	64
Kernel Size	5	5	7
Learning Rate	1e-06	1e-06	1e-06
Track Length	50	50	100

Table 4.24: Hyper-parameters of 3 harmonic embedding CNN models with binary classification targets.

Model	1	2	3
Accuracy	0.23	0.23	0.16
MIREX	0.33	0.32	0.24

Table 4.25: Results of harmonic embedding CNN on training set with binary classification targets.

Model	1	2	3
Accuracy	0.23	0.24	0.16
MIREX	0.30	0.29	0.25

Table 4.26: Results of harmonic embedding CNN on validation set with binary classification targets.

4.4 Result of Best Models on Test Set

All models in tables 4.27 trained of one-hot encoded input.

Model	Best NHCNN	Best HCNN	Best NHLSTM	Best HLSTM
Accuracy	0.52	0.55	0.45	0.52
MIREX	0.59	0.63	0.52	0.58

Table 4.27: Results of best CNN and LSTM models on validation set.

Model	Best NHCNN	Best HCNN	Best NHLSTM	Best HLSTM
Accuracy	0.50	0.55	0.45	0.52
MIREX	0.58	0.62	0.52	0.58

Table 4.28: Results of best CNN and LSTM models on test set.

4.5 Analysis

Given the results in tables 4.27 and 4.28, the best model overall is the harmonic CNN with categorical classification targets. It achieves an accuracy of 55% and MIREX score of 62% on the test set. In fact, the CNN models consistently achieve better results than the LSTM models. The HCNN beats the Temperley baseline by 13% accuracy and 18% MIREX. The other well-performing CNN models beat the Temperley baseline by over 8% accuracy and 13% MIREX. The LSTM models reach an accuracy of similar to the baseline. Their MIREX scores beat the baseline by at least 7%. The well-performing models are better at capturing similar keys than the Temperley baseline.

The difference between accuracy and MIREX score speaks of how a model misclassifies a key. A bigger difference means that a misclassified key is related to the target key by a relation encoded in the MIREX score: dominant, relative or parallel. A smaller difference means the model either classifies the model correctly or mistakes it for an unrelated key. For all models, the difference between accuracy and MIREX is between 6% and 9%.

Using the binary classification targets does not improve model performance. Binary classification outputs work much better for models with learned embeddings but still do not outperform categorical classification models. Following the MIREX score, harmonic models generally perform better than non-harmonic models with learned embeddings.

It is important for a model to be consistent across datasets. The validation and test results are similar for all four best models reported in the previous section. The performance is quite consistent between training and other sets. The difference is 5% or less for the CNN models and around 10% for the LSTM models. The LSTM models overfit more than the CNN models. The CNN models perform best with sequences of 200 notes. The LSTM perform best with shorter sequences of 50 notes. This can be a factor when choosing which model to use, depending on the data available.

Chapter 5

Error analysis

5.1 Distribution of predictions

Figure 5.1 illustrates the frequency with which each method - the HCNN, HLSTM and modified LHS - predicted each of the possible target keys as well as the distribution of keys in the complete dataset. The LHS algorithm is the most faithful to the target key distribution. The HCNN virtually ignores minor keys as they are less common in the dataset. Both the HCNN and HLSTM vastly overestimate the presence of B-major in the dataset. The LHS algorithm overestimates the presence of E minor. The distributions predicted by HCNN and HLSTM are quite similar.

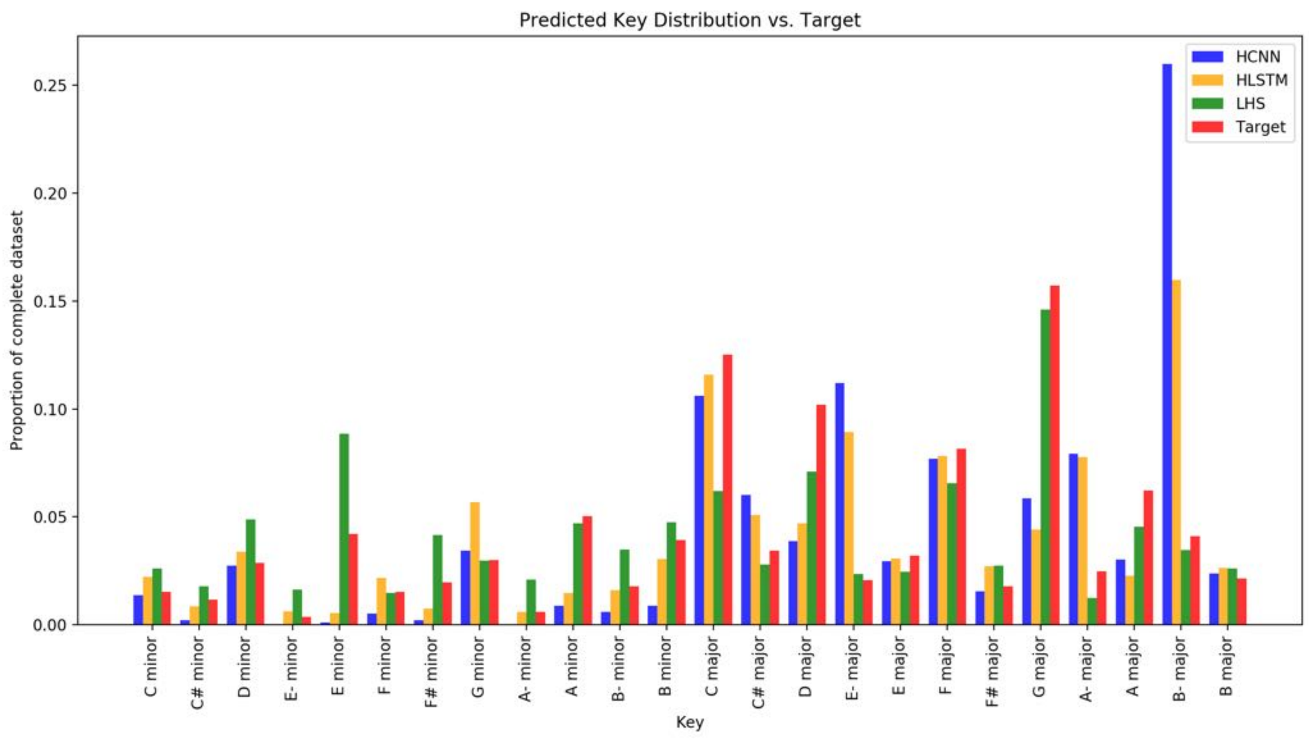


Figure 5.1: Distribution of keys predicted per model and the distribution of target keys over the complete dataset.

5.2 t-SNE embedding of predictions vs. target keys

The t-SNE plots show a 2-dimensional embedding of the 24-dimensional probability distribution over keys output by the best harmonic and non-harmonic CNN and LSTM using the test dataset. We use the test set rather than the entire dataset to reduce the t-SNE computation time. For each network, one plot is marked with the predicted keys, the other is marked with the target keys. These plots show that the networks see the key space as quite clean, representing the keys as relatively distinct. The key placement shows which keys the network sees as more related than others. The plot marked with the target keys shows that the target key space is a lot more ambiguous than that. We can see which keys tend to be confused and where the network has learned the data is all of one key, when in fact it is a complete mixture.

We notice that for all models, the order of the keys approximately follows that of the circle of fifths, see Figure 5.6. The circle of fifths is useful and informative for

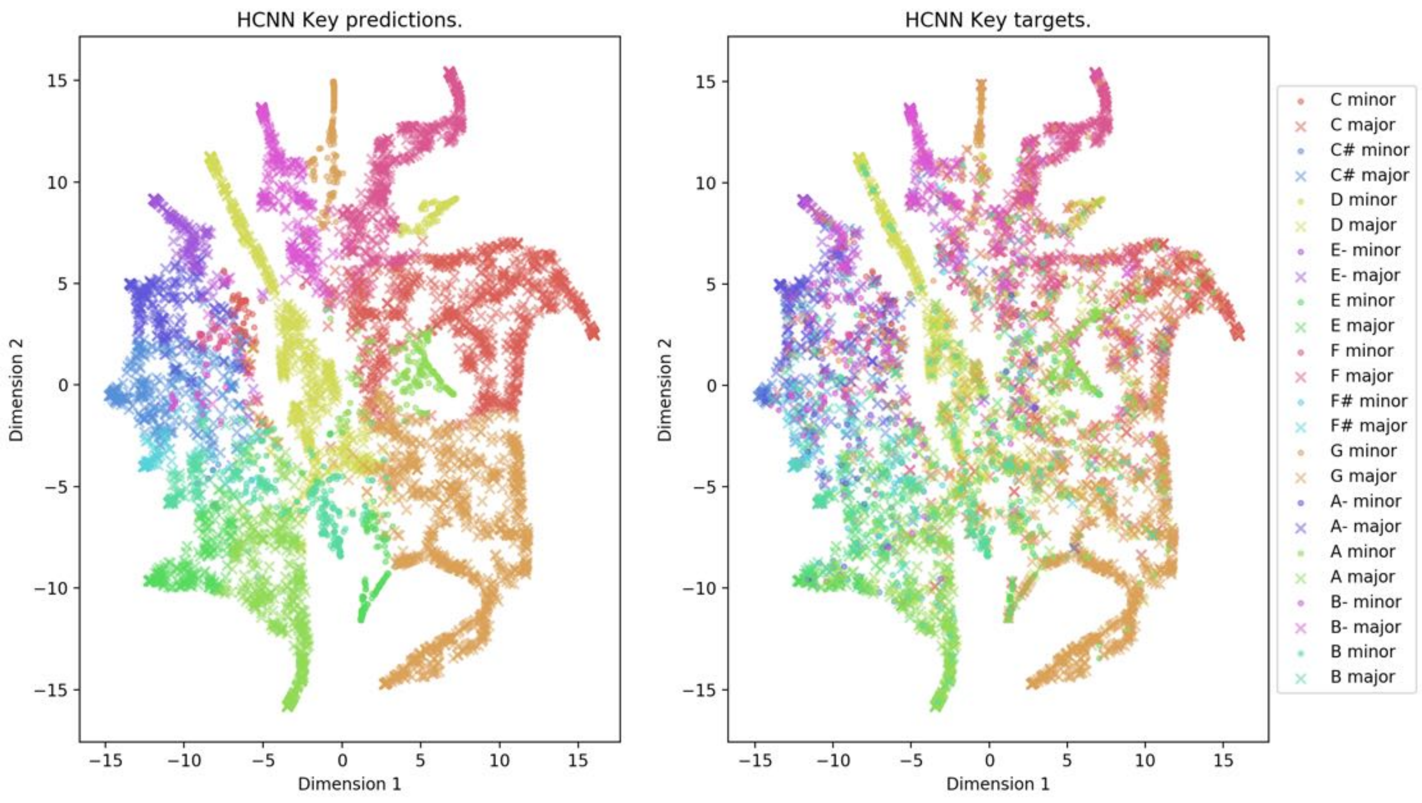


Figure 5.2: t-SNE embedding of HCNN predictions labelled with predictions and targets over the test set.

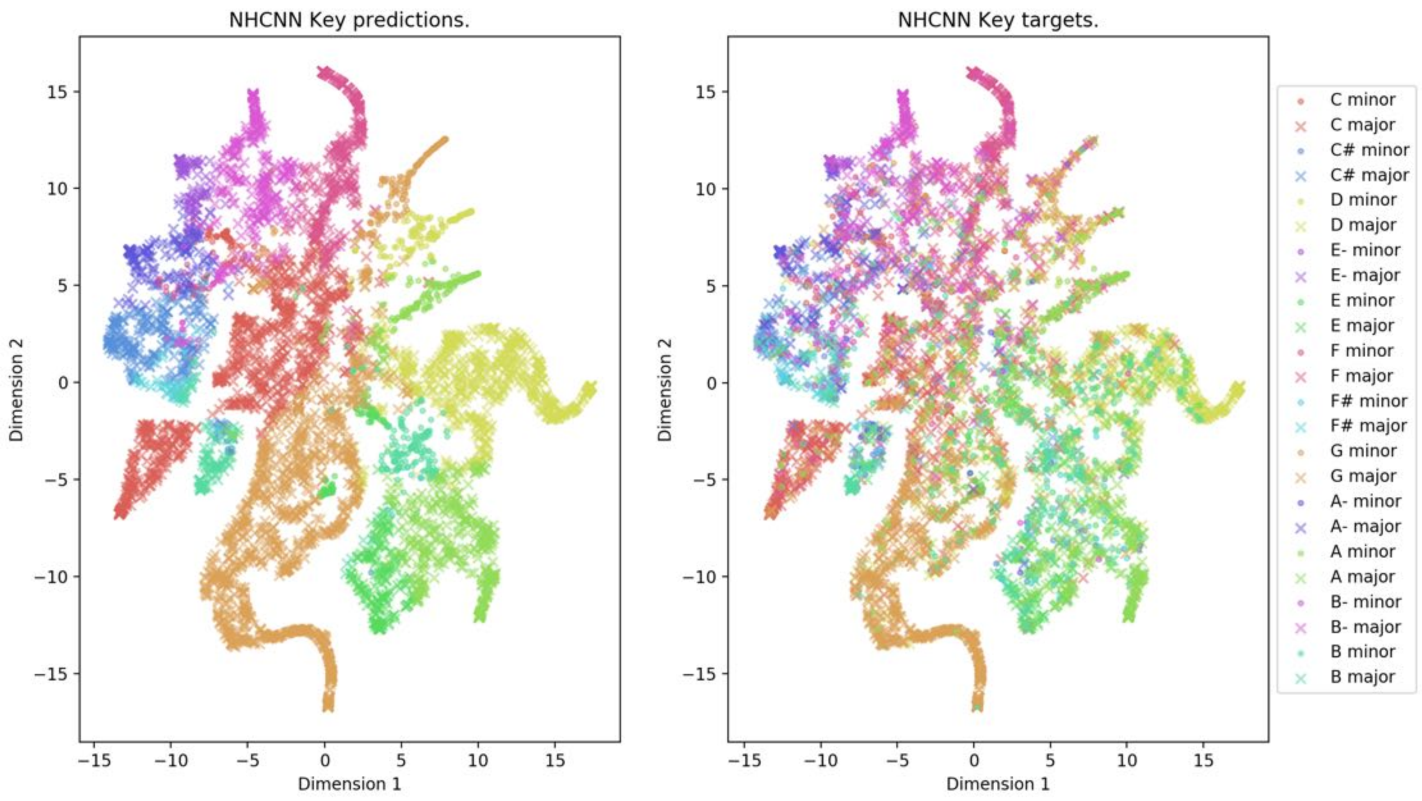


Figure 5.3: t-SNE embedding of NHCNN predictions coloured with predictions and targets over the test set.

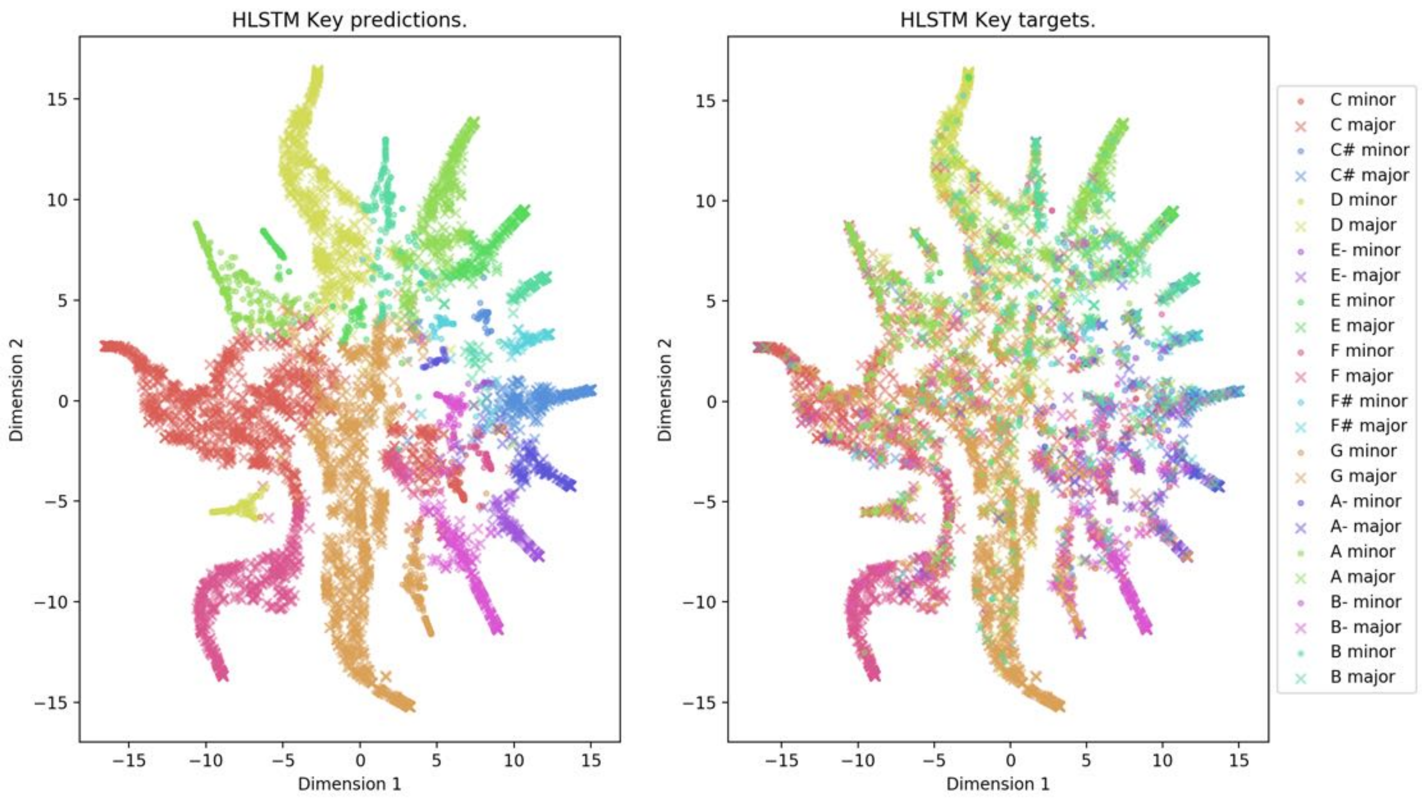


Figure 5.4: t-SNE embedding of HLSTM predictions coloured with predictions and targets over the test set.

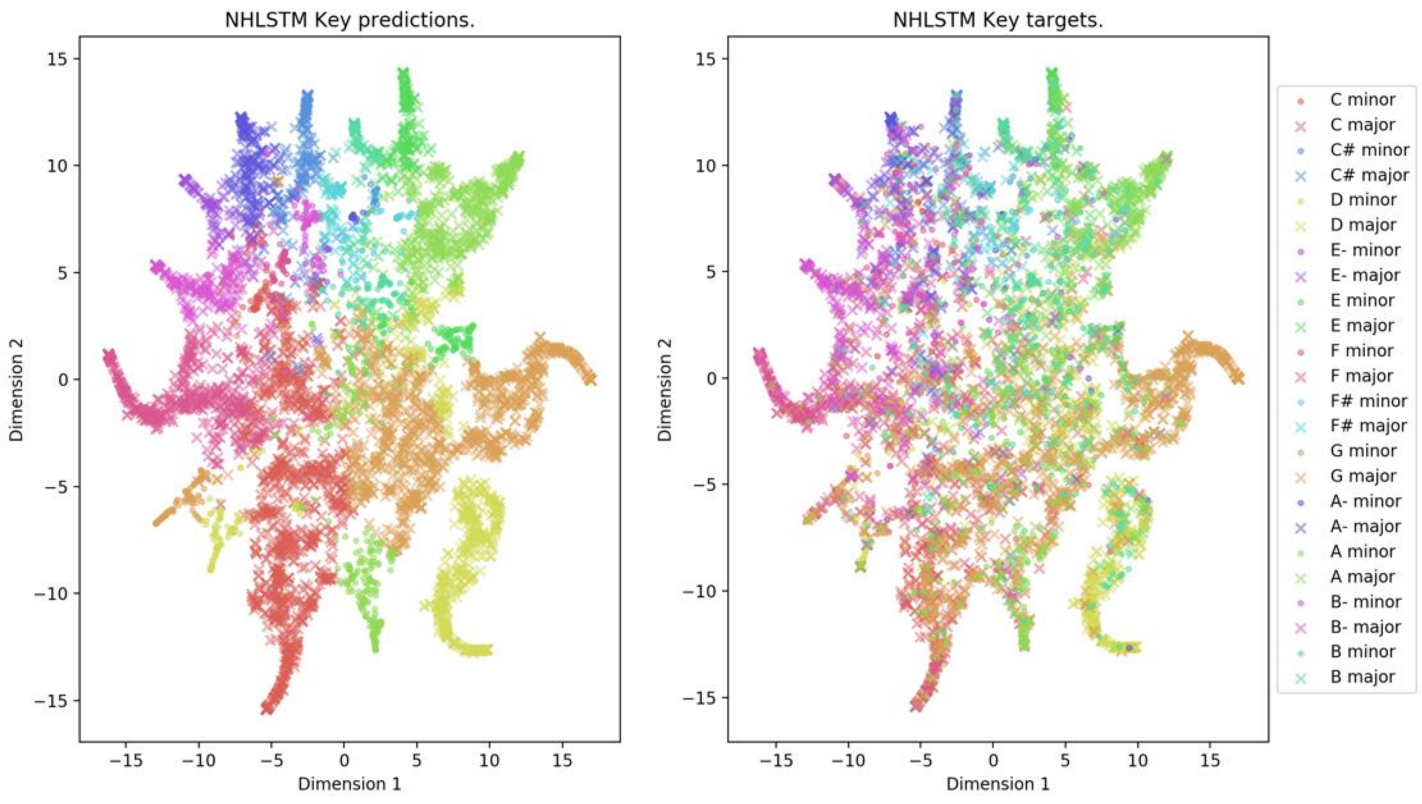


Figure 5.5: t-SNE embedding of NHLSTM predictions coloured with predictions and targets over the test set.

a musician to understand the relationship between keys. The outer circle shows the key signatures, the middle circle shows the major keys, and the inner circle shows the minor keys. The minor key in the same slice as a major key is that major key's relative minor and vice-versa. One step clockwise brings the key up a fifth. One step counter-clockwise is the key down one fifth or up one fourth. Composers often use the keys next to the main key of the piece as per this circle to find good-sounding modulations. For example, a song in C would welcome modulations in F and G, as these three keys have overlapping chords. The HLSTM shows the closest resemblance to the circle of fifths, see Figure 5.4.

Aside from this exciting discovery, the LSTM models have clearer representations of minor keys, whereas the minor points in the CNN t-SNE plots tend to blend into major key clusters. The HLSTM shows tighter clusters than the non-harmonic LSTM (NHLSTM), while the HCNN displays clusters that are more spread out than its non-harmonic counterpart. Comparing the key targets to their predictions, it seems that the major keys are generally well-predicted. For all models, tracks with minor keys appear all over the plots of key targets, losing the neat definition the model predicted. The clusters corresponding to minor keys are much smaller than those of major keys. It is possible a balanced dataset with the same number of tracks per key would lead to better defined clusters for all keys.

5.3 Comparison of learned embedding vs. harmonic embedding

The embeddings learned by the NHCNN and NHLSTM are reasonable from a music theory standpoint (Figure 5.7). The NHCNN displays the notes similarly to a squashed circle of fifths, see Figure 5.6. The embedding learned by the NHLSTM looks like a cross with C at its centre. Each branch of the cross is made of notes within a perfect fifth of each other. The embeddings both appear to agree on the following note groupings: C at the centre, [F#, C#, A#, E-], [A, D, G], [B, E] and [B-, F]. These groups all contain notes that follow each other on the scale of perfect fifths, information also encoded in the harmonic network. It in not obvious whether the non-harmonic embeddings learned to encode any relation between notes separated by a major third, as in the harmonic network.

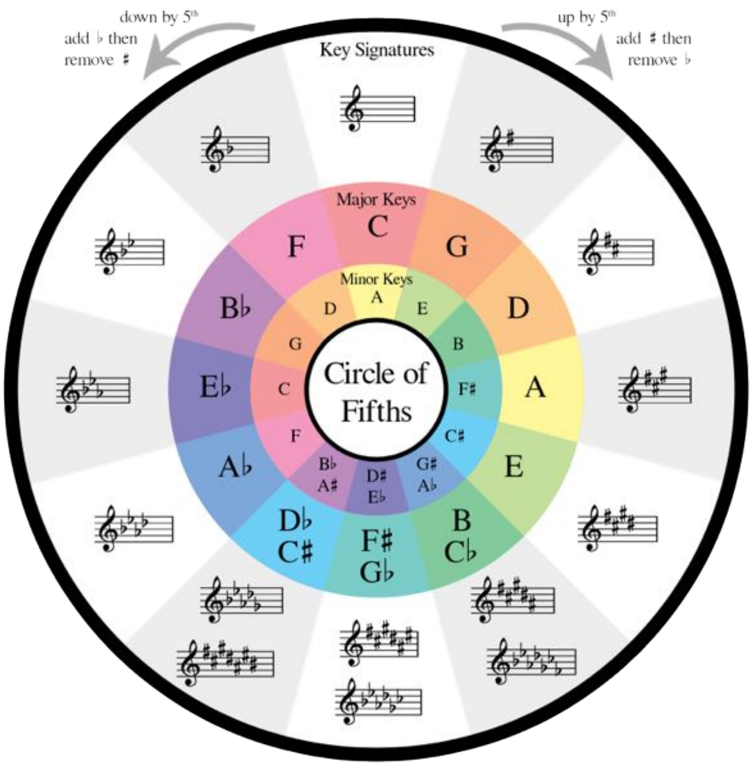


Figure 5.6: Circle of fifths.

5.4 Well-Tempered Clavier error analysis

To explore the typical errors made by the neural networks, we analyse the predictions HCNN and HLSTM made on the 48 fugues of the Well-Tempered Clavier. We also include the predictions made by the LHS algorithm for comparison.

The probability heat maps in Figures 5.8 and 5.9 show how confidently the network predicted the key for each track in the Well-Tempered Clavier. This level of confidence can then be linked to the strength of layer activations for each note in these tracks later on, as well as to the key distributions shown earlier. We use these heat maps in tables 5.2 and 5.3 to determine alternative keys the models could have predicted.

According to Tables 5.1, 5.2 and 5.3, the modified LHS algorithm has an accuracy of 83% on the Well-Tempered Clavier. The HLSTM achieves 73% and the HCNN 75%. The original LHS algorithm was designed for this dataset, which explains why it outperforms the neural networks which outperformed it on the complete dataset.

The errors made by the three different models overlap each other. They all fail to identify that Fugue 14 of Book II is F# minor. The modified LHS and HCNN both identify the fugue as A major, the target's relative key. The HLSTM identifies it as B minor, the subdominant. The modified LHS and the HCNN confuse Book II fugue 18

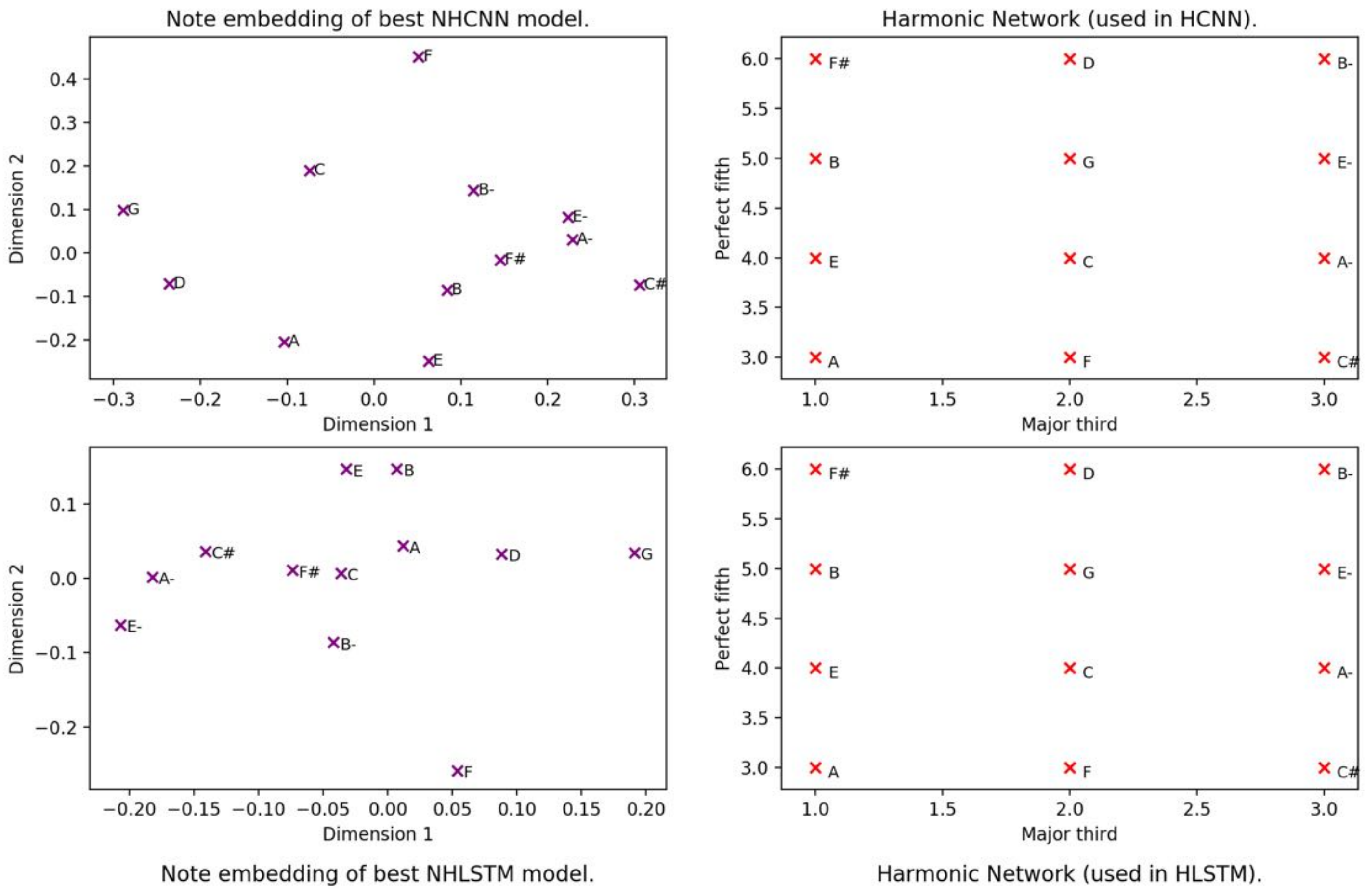


Figure 5.7: Non-harmonic and harmonic embeddings for both CNN and LSTM.

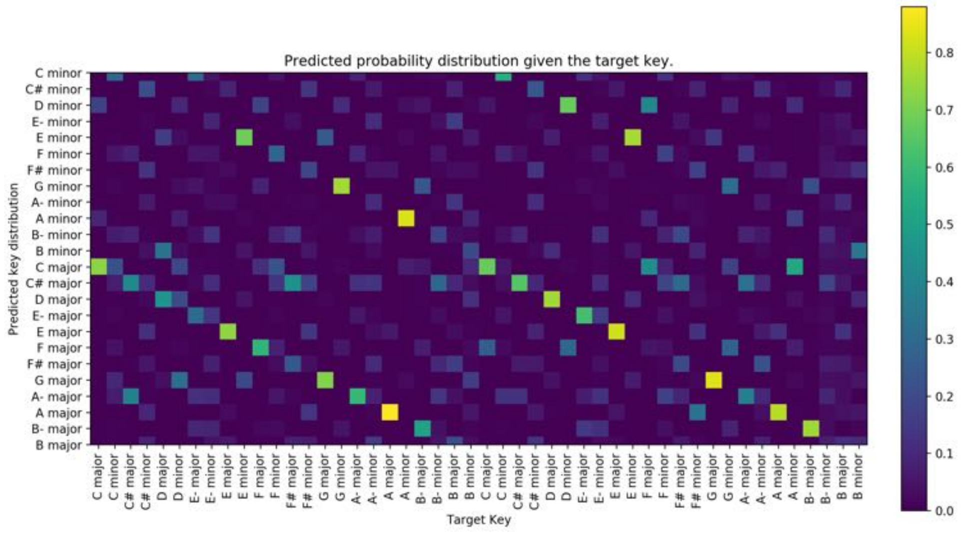


Figure 5.8: Heat map of HCNN probability vector outputs for the Well-Tempered Clavier dataset for target key versus predicted keys.

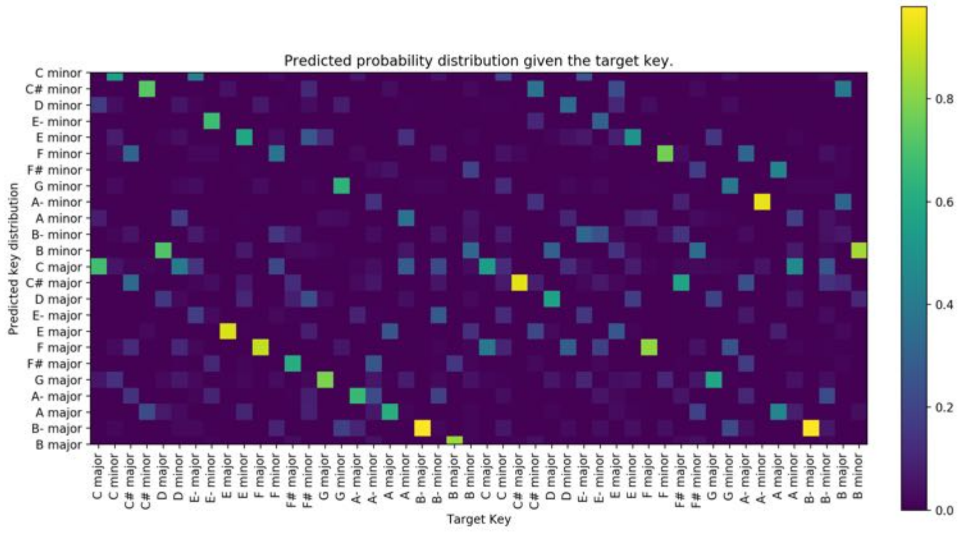


Figure 5.9: Heat map of HLSTM probability vector outputs for the Well-Tempered Clavier dataset for target key versus predicted keys.

in A- minor as B major and F# major respectively. B major makes more sense than F# major as it is the key relative to A- minor. The HLSTM and HCNN both misidentify Book II fugue 13 in F# major as C# major, its dominant. These two models also misidentify Book II fugue 20 in A minor and fugue 22 in B- minor as C major. C major is the dominant of the A minor. There is no significant relationship between B- minor and C major. The confidence heat maps in Figures 5.8 and 5.9 show very low confidence in any prediction for a fugue in B- minor. Figure 5.1 shows that B- minor is scarcely represented in the dataset. We can assume that, given not much information about this key, the models choose to assign a very popular key to the fugue.

Fugue	**Prediction**	**Target**	**Notes**
Book I Fugue 2	C major	C minor	Note 2 is B, pointing to C major and excluding C minor. C major is the parallel of C minor.
Book I Fugue 20	A major	A minor	Note 2 is G#, pointing to A major, excluding A minor. A minor and major are parallel keys.
Book II Fugue 2	G minor	C minor	The sixteenth note is A, excluding all keys but G minor. G minor is the dominant of C minor.
Book II Fugue 3	F# major	C# major	The note B causes misidentification. F# major is the subdominant of C# major.
Book II Fugue 8	E- major	E- minor	Note 4 is D which excludes the true key E- minor in favour of its parallel E- major.
Book II Fugue 14	A major	F# minor	A major is the relative major of F# minor, so consisting of the same set of notes. The track begins with A, misidentified as the tonic of A major.
Book II Fugue 18	B major	A- minor	Keys are relative to each other. Track begins with tonic of B major.
Book II Fugue 24	F# minor	B minor	Fourth note is B- excluding all keys before we get to G which would point to B minor. Instead the tonic-dominance rule points to F# minor because the track begins with F#. F# minor is the dominant of B minor.

Table 5.1: LHS algorithm errors on the fugues of the Well-Tempered Clavier.

Fugue	Prediction	Alternative	Target	Notes
Book I Fugue 5	B minor	D major	D major	B minor is the relative minor.
Book I Fugue 6	C major	A minor	D minor	A minor is the dominant of D minor. C major is the relative major of A minor.
Book I Fugue 7	C minor	E- major	E- major	C minor is the relative key.
Book I Fugue 14	E minor	D major	F# minor	No significant relationship between prediction and target.
Book I Fugue 18	F# major	A- major	A- minor	Third highest probability is A- minor. No significant relationship between F# major and target. A- major is parallel major of target.
Book I Fugue 22	E- major	C major	B- minor	E- major is the relative major of the subdominant of the target. C major is very frequent in the dataset.
Book II Fugue 7	B- minor	C minor	E- major	C minor is the relative minor of the target.
Book II Fugue 13	C# major	B- minor	F# major	C# major is the dominant of the target.
Book II Fugue 14	B minor	F# minor	F# minor	B minor is the subdominant of the target.
Book II Fugue 17	F minor	C# major	A- major	F minor is the relative minor of A- major.
Book II Fugue 20	C major	A minor	A minor	C major is the dominant of A minor.
Book II Fugue 22	C major	A- major	B- minor	No significant relationship. C major is very frequent in the dataset.
Book II Fugue 23	C# minor	A- minor	B major	C# minor is the relative minor of E major, the subdominant of the target. A- minor is enharmonic with G# minor, the relative minor to the target.

Table 5.2: HLSTM errors on the fugues of the Well-Tempered Clavier.

Fugue	Prediction	Alternative	Target	Notes
Book I Fugue 6	G major	D major	D minor	D major is the parallel of D minor. G major is the subdominant of D major.
Book I Fugue 8	E- major	C# major	E- minor	E- major is the parallel of E- minor.
Book I Fugue 13	C# major	F# major	F# major	C# major is the dominant of F# major.
Book I Fugue 18	B major	C# major	A- minor	B major is enharmonic with C flat major, the relative key of A- minor.
Book I Fugue 22	C# major	B- minor	B- minor	C# major is enharmonic with D flat major, the key relative to B- minor.
Book II Fugue 11	C major	D minor	F major	C major is the dominat to F major. D minor is the key relative to F major.
Book II Fugue 12	C# major	F# major	F minor	C# major is the key relative to B flat minor, the subdominant of F minor. F# major is the subdominant of B flat minor.
Book II Fugue 13	C# major	F# major	F# major	C# major is the dominant of the target.
Book II Fugue 14	A major	F# minor	F# minor	F# minor is relative to A major.
Book II Fugue 18	F# major	C# major	A- minor	F# major is the subdominant of C# major. C# major is the subdominant of G# major, enharmonically equivalent with A- minor.
Book II Fugue 20	C major	A minor	A minor	C major is the relative key to A minor.
Book II Fugue 22	C major	A- major	B- minor	No significant relationship. C major is very frequent in the dataset.

Table 5.3: HCNN errors on the fugues of the Well-Tempered Clavier.

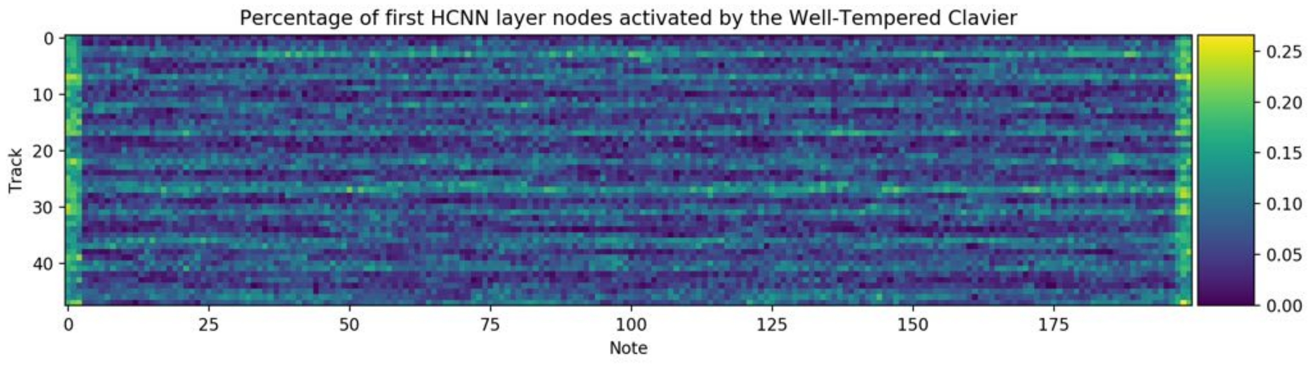


Figure 5.10: Percentage of first HLSTM layer nodes activated by the Well-Tempered Clavier.

5.5 Layer activations

The layers of a neural network can be visualised as shown in Simonyan et al. (2013), Karpathy et al. (2015).

The activations of a neural network can be visualised to observe what the network reacts to. It is increasingly common in computer vision to see what the network uses to classify an image. The deeper the layer, the more abstract the feature it recognises. We will study the first layer of which typically shows the most recognisable features. In this section, we will compare the activations of the first layer of the HCNN and HLSTM. For each note in the fugue, we track the proportion of nodes in the layer whose activation is over 0.4. In the figures below, the more nodes activated for each note, the lighter the colour. The LSTM is known for learning sequences and the CNN is known for learning patterns invariant to translation. We hope to see such patterns reflected in the layer activations.

Looking at the plot of activations in Figures 5.10 and 5.11, we see that the HCNN

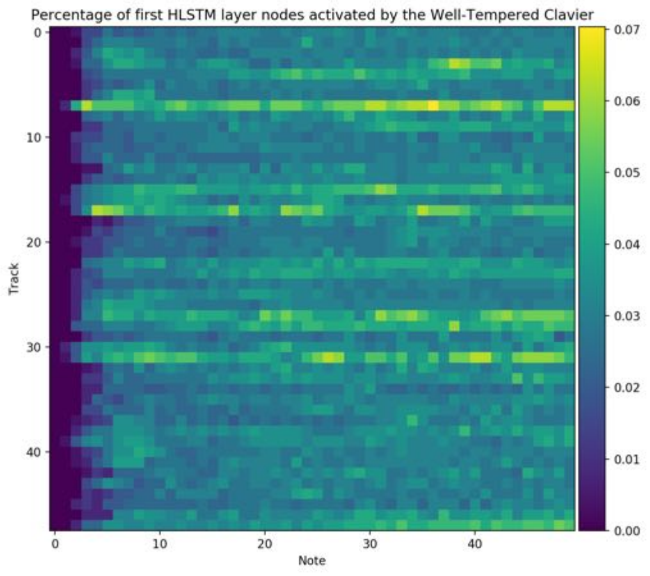


Figure 5.11: Percentage of first HLSTM layer nodes activated by the Well-Tempered Clavier.

reacts more strongly to individual notes, with up to 25% of 128 nodes activating for a single note. The HLSTM activates a lower percentage of nodes of larger sequence of notes, up to 7% of 256 activating for sequence of notes. This fits with the previous ideas of these two models.

Analysing activations per note as displayed in Figures 5.12 and 5.13, it appears that both methods have made the connection between key and the notes in that key scale. If the models make a mistake, it is often because the sequence of track they are given does contain a lot of a note that is not the tonic of the track's key. The fugue in D minor misidentified as G major by the HCNN activates nodes for many occurrences of F# and C#, two accidentals to D minor, see 5.12. The scale of G major contains F# and is much more common in the dataset. Because we have truncated the tracks to either 200 or 50 notes, it is possible the resulting track contains a modulation, with a key different to the track's main key. The HLSTM predicts a fugue F# major as C# major. We can see in Figure 5.13 that the model activated for a sequence containing many C#. C# major is the dominant of F# major, but much more common in the dataset. So, if it isn't that the truncated sequence in question contains a modulation to C# major, it is likely that an undecided model will favour the most common key.

Contrary to the modified Longuet-Higgins Steedman method, the neural networks do not place high importance on the first notes of the piece. The HLSTM, as seen in

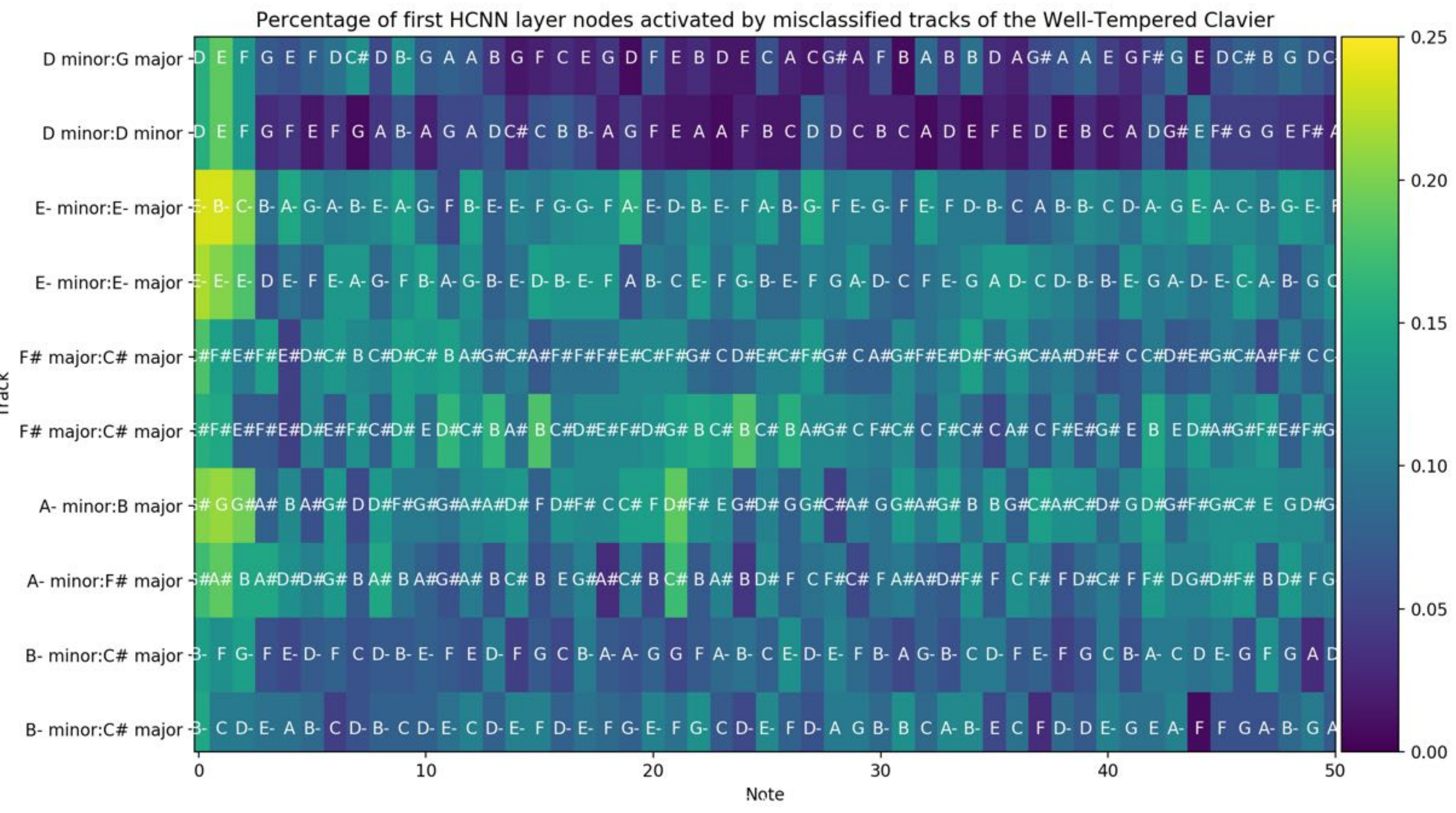


Figure 5.12: A closer look at the HCNN activations of misclassified tracks identified as Target:Prediction.

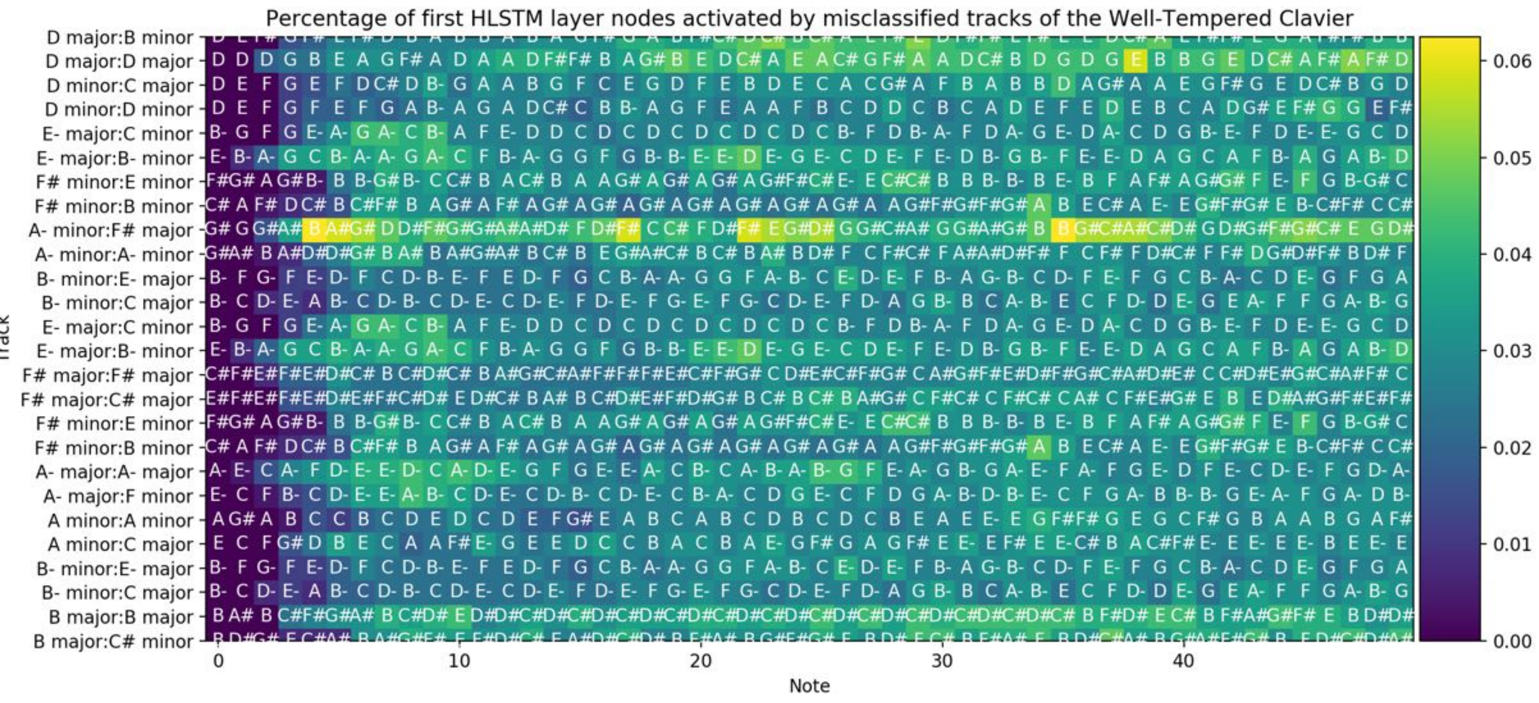


Figure 5.13: A closer look at the HLSTM activations of misclassified tracks identified as Target:Prediction.

the Figures mentioned above, does not show any noticeable activation for the first few notes in each track. The HCNN does show strong activation of up to 25% on the first few notes. In the sample in Figure 5.12, the first note of the tracks is not the tonic of the key they have predicted. This tells us the first note contributes to the decision, but without a such direct correspondence as the tonic-dominant rule.

Chapter 6

Conclusion

6.1 Model performance and interpretation

We successfully created LSTM and CNN models which outperformed the baseline methods. We found that CNNs achieve the best results, meaning that memorising sequences as done by an LSTM is not an advantage over the pattern recognition performed by a CNN. The harmonic embedding performed better than learned embedding, meaning incorporating music theory into a model does improve performance. When studying the learned embedding, we noticed the embedding was capturing the similar information to the harmonic embedding. This indicates that the notions of major third, perfect fifth and generally the distance between notes are in fact important to key identification. Categorical classification models with targets assuming all keys are distinct outperformed models which attempted to encode relationships between the targets using binary classification. This could be because the binary classification assumed the notes were independent from each other, which the categorical classification did not. The t-SNE embeddings of predictions for all models demonstrated that a model unsupervised will organise keys in the same order as the circle of fifths. Further analysis on the Well-Tempered Clavier revealed that the models do rely on the presence of notes in the track to determine the key. This corresponds to the human idea of key scale. What's more, the errors made by the models on the Well-Tempered Clavier were often to a key whose relationship is close to the target. These findings point to the conclusion that, without needing to be taught, a neural network learns the same music theory as a human.

6.2 Further Work

The models could be improved by testing different architectures. For example, the LSTM could be combined with the CNN, the layer of the LSTM could be made bi-directional and the CNN could include more pooling layers. The binary classification model could be turned into an encoder-decoder network, where a neural network would determine the key from the binary classification output, instead of using the total probability rule. Instead of truncating the data and discarding the rest of the track, we could augment the data by segmenting each track and considering each segment as a separate track.

We could also improve the key bias in the data by adding more tracks in minor key or else weighting the data. Further analysis could also be performed to determine the performance of each model by genre to check for bias toward one genre and investigate how different keys are represented across genres.

Lastly, the music representations gathered throughout this project to research the relationship between the key and the mood of a track. The Lakh dataset is matched to the Million Song Dataset, which itself contains a multitude of properties for each track. Possible characteristics to include in later work are those related to feeling, such as danceability, energy and loudness.

Appendix A

A.1 Pseudocode for the modified Longuet-Higgins Steedman algorithm

Algorithm 1 modified Longuet-Higgins and Steedman algorithm

```
for note in track do
    possible keys = keys that include note
    if only one key remaining then
        return key
    else if two keys remaining then
        if keys are relative then
            run tonic-dominant preference rule on these keys
            if solution found then
                return key
            else
                return both keys
            end if
        end if
    else if no keys remaining then
        run tonic-dominant preference rule on previously remaining keys
        if more than one key passes the tonic-dominant preference test then
            possible keys = the keys that passed the test
            move on to next note and continue key search
        else if one key found then
            return key
        end if
    else
        continue
    end if
end for
if more than one key remaining then
    run tonic dominant preference rule
    return key
end if
```

Bibliography

Baroni, Mario. A grammar for melody: Relationships between melody and harmony. *Musical Grammars and Computer Analysis, Florence*, pp. 201–218, 1984.

Baroni, Mario, Maguire, Simon, and Drabkin, William. The concept of musical grammar. *Music Analysis*, 2(2):175–208, 1983.

Belinkov, Yonatan and Glass, James R. Analyzing hidden representations in end-to-end automatic speech recognition systems. *CoRR*, abs/1709.04482, 2017. URL `http://arxiv.org/abs/1709.04482`.

Belinkov, Yonatan, Màrquez, Lluís, Sajjad, Hassan, Durrani, Nadir, Dalvi, Fahim, and Glass, James. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. *arXiv preprint arXiv:1801.07772*, 2018.

Bergstra, James S, Bardenet, Rémi, Bengio, Yoshua, and Kégl, Balázs. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pp. 2546–2554, 2011.

Bertin-Mahieux, Thierry, Ellis, Daniel P.W., Whitman, Brian, and Lamere, Paul. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

Brownlee, Jason. Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras, July 2016. URL `https://machinelearningmastery.com/`.

Campbell, Spencer Evison. *Automatic Key Detection of Musical Excerpts from Audio*. PhD thesis, McGill University, 2011.

Chew, Elaine. The Spiral Array: An Algorithm for Determining Key Boundaries. *Music and Artificial Intelligence, Lecture Notes in Computer Science.*, 2445, 2002.

Colin Raffel. *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching*. PhD thesis, Columbia University, 2016.

Cuthbert, M and Ariza, C and Hogue, B and Oberholtzer, J. music21 Dataset, 2017. URL `http://web.mit.edu/music21/doc/about/about.html`.

Dalvi, Fahim, Durrani, Nadir, Sajjad, Hassan, Belinkov, Yonatan, and Vogel, Stephan. Understanding and improving morphological learning in the neural machine translation decoder. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 142–151, 2017.

Dewancker, Ian, McCourt, Michael, and Clark, Scott. Bayesian optimization primer, 2015.

Euler, Leonhard. *Tentamen novae theoriae musicae ex certissismis harmoniae principiis dilucide expositae*. Saint Petersburg Academy, 1739.

Granroth-Wilding, Mark and Steedman, Mark. A Robust Parser-Interpreter for Jazz Chord Sequences. *Journal of New Music Research*, 43(4):355–374, 2014.

Graves, Alex. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

Hamel, Philippe and Eck, Douglas. Learning features from music audio with deep belief networks. In *ISMIR*, volume 10, pp. 339–344. Utrecht, The Netherlands, 2010.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Karpathy, Andrej, Johnson, Justin, and Fei-Fei, Li. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

Korzeniowski, Filip and Widmer, Gerhard. End-to-End Musical Key Estimation Using a Convolutional Neural Network. In *25th European Signal Processing Conference*. EURASIP, 2017a.

Korzeniowski, Filip and Widmer, Gerhard. End-to-end musical key estimation using a convolutional neural network. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pp. 966–970. IEEE, 2017b.

Krumhansl, Carol L. *Cognitive Foundations of Musical Pitch.* Oxford University Press.

Krumhansl, Carol L. *Cognitive foundations of musical pitch*, volume 17. Oxford University Press, 2001.

Li, Jiwei, Chen, Xinlei, Hovy, Eduard H., and Jurafsky, Dan. Visualizing and understanding neural models in NLP. *CoRR*, abs/1506.01066, 2015. URL `http://arxiv.org/abs/1506.01066`.

Long, Jonathan, Shelhamer, Evan, and Darrell, Trevor. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440. IEEE, 2015.

Longuet-Higgins, H.C. and Steedman, Mark. *On Interpreting Bach*, pp. 221–241. MIT Press, 1987.

Maaten, Laurens van der and Hinton, Geoffrey. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Mardirossian, Arpi, Chuan, Ching-Hua, and Chew, Elaine. 2005:Audio and Symbolic Key, 2005. URL `http://www.music-ir.org/mirex/wiki/`.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.

Sapp, Craig Stuart. Visual Hierarchical Key Analysis. *Comput. Entertain.*, 3 (4):1–19, oct 2005. ISSN 1544-3574. doi: 10.1145/1095534.1095544. URL `http://doi.acm.org/10.1145/1095534.1095544`.

Schreiber, Hendrik. CNN-Based Automatic Musical Key Detection. *Music Information Retrieval Evaluation eXchange*, 2017.

Shahriari, Bobak, Swersky, Kevin, Wang, Ziyu, Adams, Ryan P, and De Freitas, Nando. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

Shuvaev, Sergey, Giaffar, Hamza, and Koulakov, Alexei A. Representations of sound in deep learning of audio features from music. *arXiv preprint arXiv:1712.02898*, 2017.

Simonyan, Karen, Vedaldi, Andrea, and Zisserman, Andrew. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Sprengel, Elias, Jaggi, Martin, Kilcher, Yannic, and Hofmann, Thomas. Audio based bird species identification using deep learning techniques. Technical report, 2016.

Strobelt, Hendrik, Gehrmann, Sebastian, Huber, Bernd, Pfister, Hanspeter, and Rush, Alexander M. Visual analysis of hidden state dynamics in recurrent neural networks. *CoRR*, abs/1606.07461, 2016. URL `http://arxiv.org/abs/1606.07461`.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014.

Temperley, David. What's Key for Key? The Krumhansl-Schmuckler Key-Finding Algorithm Reconsidered. *Music Perception: An Interdisciplinary Journal*, 17(1): 65–100, 1999.

Temperley, David and Tan, Daphne. Emotional connotations of diatonic modes. *Music Perception: An Interdisciplinary Journal*, 30(3):237–257, 2013.

Turian, Joseph, Ratinov, Lev, and Bengio, Yoshua. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pp. 384–394. Association for Computational Linguistics, 2010.

van der Westhuizen, Jos and Lasenby, Joan. Visualizing lstm decisions. *stat*, 1050:23, 2017.

Vinyals, Oriol, Toshev, Alexander, Bengio, Samy, and Erhan, Dumitru. Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555, 2014.

Zaremba, Wojciech, Sutskever, Ilya, and Vinyals, Oriol. Recurrent neural network regularization. *CoRR*, abs/1409.2329, 2014.

Zhang, Ning, Donahue, Jeff, Girshick, Ross, and Darrell, Trevor. Part-Based R-CNNs for Fine-Grained Category Detection. In *European Conference on Computer Vision*, pp. 834–849. Springer, Cham, 2014.