

Ein Erklärungsansatz für CNN-basierte Musik Genre Klassifikationssysteme mittels semantischer Deskriptoren

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Lukas Rotheneder, BSc

Matrikelnummer 01529223

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Andreas Rauber

Wien, 14. März 2023

Lukas Rotheneder

Andreas Rauber

An Explanation System for CNN Music Genre Classification based on Semantic Descriptors

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

Lukas Rotheneder, BSc

Registration Number 01529223

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Andreas Rauber

Vienna, 14th March, 2023

Lukas Rotheneder

Andreas Rauber

Erklärung zur Verfassung der Arbeit

Lukas Rotheneder, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 14. März 2023

Lukas Rotheneder

Acknowledgements

I want to thank my supervisor Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Andreas Rauber for the opportunity to work on such an exciting topic and the attentive supervision throughout the entire period of writing my master thesis. Additionally, I want to thank Univ.Lektor Dipl.-Ing. Dr.techn. Alexander Schindler for assisting with his expertise and leading me on the right track, and my girlfriend Lena for the helpful suggestions, the emotional support and the understanding in stressful times.

Kurzfassung

Convolutional Neural Networks (CNN) erzielen gute Ergebnisse bei der Klassifizierung von Musikgenres, sind aber nicht ohne weiteres interpretierbar. Diese Masterarbeit stellt einen Erklärungsansatz vor, der die Semantik beschreibende Musik Eigenschaften als Rechtfertigung für eine Genreklassifizierung liefert, sowie einen Ansatz, um das introspektive Verständnis eines CNN Musikgenre-Klassifikationsmodells zu verbessern, indem die von den Faltungsschichten erzeugten Feature-Maps zu jenen Musik Eigenschaften zugewiesen werden, welche sie am besten abbilden. Für den Rechtfertigungsansatz wird eine systematische Literaturrecherche durchgeführt, um ein geeignetes Modell zur Vorhersage von Musik Eigenschaften auswählen zu können. Wir fanden heraus, dass semantische Musik Eigenschaften aus Audio-Dateien erlernt werden können, allerdings reicht die Anzahl der vorhergesagten Eigenschaften nicht aus, um eine Genreklassifizierung rechtfertigen zu können. Außerdem haben wir festgestellt, dass die Erklärungen in Form von semantischen Eigenschaften, die von dem vorgeschlagenen Erklärungssystem geliefert werden, für das Ground-Truth Genre relevant sind. Darüber hinaus stellen wir eine Möglichkeit vor, die bereitgestellten Eigenschaften im Modell-Input zu visualisieren. Für den zweiten Ansatz testen wir k-Nearest Neighbour und Random Forest, um semantische Eigenschaften auf Feature-Maps abzubilden. Wir bieten auch eine Visualisierung der Feature-Maps welche die semantischen Eigenschaften von zwei Beispiel Liedern am besten darstellen. Weiters wird gezeigt, wie sich Feature-Maps von unterschiedlichen Schichten unterscheiden, und bewertet, welche und wie gut die semantischen Eigenschaften von den jeweiligen Feature-Maps abgebildet werden können. Wir waren jedoch nicht in der Lage, Feature-Maps zu finden, die alleine eine semantische Eigenschaft vollständig abbilden kann.

Abstract

Convolutional Neural Networks (CNN) achieve a good performance in music genre classification but are not readily interpretable. This master thesis proposes an explanation approach that provides semantic descriptors as a justification for a music genre classification as well as an approach to gain introspective understanding of a CNN music genre classification model, by assigning appropriate semantic descriptors to feature maps, generated by the convolutional layers. For the justification approach a systematic literature review is conducted to find a suitable model to predict semantic descriptors. We found that semantic descriptors can be learned from audio inputs as used for state-of-the-art CNN genre classification models, however the number of predicted tags is not enough to justify a genre classification. Additionally, we found that the explanations in form of semantic descriptors provided by the proposed explanation system are indeed relevant to the ground truth genre. Furthermore, we present a way to visualize the provided descriptors in the input. For the introspective approach, we consider k-Nearest Neighbour and Random Forest to map descriptive labels to feature maps. We also provide a visualization of the feature maps that represents the semantic descriptors best for two example songs. We show how feature maps differ from different layers and assess how well a feature map covers the semantic of a descriptor. However, we were not able to find feature maps that alone can fully represent a semantic descriptor.

Contents

Kurzfassung	ix
Abstract	xi
Contents	xiii
1 Introduction	1
1.1 Motivation	1
1.2 Aim of the Work	2
1.3 Thesis Structure	3
1.4 Related Work	3
1.5 Convolutional Neural Networks (CNN)	4
1.6 Signal Processing	5
2 Explanation Approaches	7
2.1 Explanation System Approach with Semantic Descriptors	7
2.2 Assigning Descriptive Labels to Feature Maps generated by Convolutional Layers	8
2.3 Datasets	10
3 Model to be Explained	13
3.1 Data Pre-processing	13
3.2 CNN Architecture following [CKN ⁺ 19]	14
3.3 CNN Architecture following [PG20]	15
3.4 Implementation Details	16
3.5 Result	17
4 Learning Semantic Descriptors - Systematic Literature Review	21
4.1 Review Protocol	21
4.2 Search	23
4.3 Result	24
5 Explaining Genre Classification with Semantic Descriptors	29
5.1 Learning Semantic Descriptors	29
	xiii

5.2	Mapping between Descriptors and Genre	48
5.3	Visualization of Descriptors	51
5.4	Workflow and Evaluation	51
6	Assigning Semantic Descriptors to Feature Maps	55
6.1	Feature Maps	55
6.2	Data	57
6.3	Model Selection	58
6.4	Results	58
6.5	Evaluation	64
7	Conclusion and Future Work	65
7.1	Conclusion	65
7.2	Future Work	66
	List of Figures	67
	List of Tables	69
	Bibliography	71
A	Cross-Validation Result for Assigning Feature Maps to Descriptors	77
A.1	Cross-Validation Result - Convolutional Layer 1	77
A.2	Cross-Validation Result - Convolutional Layer 2	77
A.3	Cross-Validation Result - Convolutional Layer 3	77

CHAPTER 1

Introduction

1.1 Motivation

Black-box models are used in more and more applications in a variety of fields, therefore explainability of such machine learning models is steadily gaining importance. This becomes even more important in fields where decisions can lead to serious consequences for certain people, like medical diagnosis, financial decision making and self-driving cars. Therefore, these models require deeper understanding not only for improving safeness by retracing wrong decisions, but also for legal reasons [ZGCH21].

By explaining decisions of black-box models, to which class most of the state-of-the-art music genre classification models belong, one can differentiate between introspection explanations, whose aim is to expound on which model specific setting the final output was determined, and justification explanations, which relate the final output of the model to separately learned tangible features to validate a decision [HAR⁺16].

Deep Neural Networks (DNN) show high effectiveness in music genre classification [CKN⁺19, JHP22, LA22, GSS20], however, their performance comes at the cost of missing interpretability of their predictions. This is one of the main problems research is currently facing [HAR⁺16]. Providing explanations for the decisions of Deep Neural Networks with visual evidence (or audible evidence in the case of audio classification) leads to more trustworthy systems [HAR⁺16]. This can be even further enhanced by grounding the generated explanations within the input [HHDA18]. Furthermore, developing an explanation system and providing explanations based on descriptive labels, such as occurring instruments or semantic descriptions of timbre and mood, would be more suitable for music experts who usually do not have experience with artificial intelligence (AI) models to work with an introspection explanation system.

To measure how accurate the generated explanations describe the genre, a clear definition is needed and therefore a mapping between semantic descriptors, provided by the

explanation systems, and genres is needed.

1.2 Aim of the Work

The aim of this work is the development of an explanation system for state-of-the-art Deep Neural Networks for genre classification. The system should provide descriptive labels for songs whose genre was predicted by a state-of-the-art model. In order to understand the semantic descriptors as an explanation of the genre classification, the descriptors should reflect properties of the given audio instance and also fit the predicted genre [HAR⁺16]. In the course of this work, we investigate if state-of-the-art models are able to learn music characteristics from audio files and if they can be used as explanation for a specific genre decision. For this purpose, a systematic literature review was conducted to determine appropriate models.

This thesis is restricted to explaining Convolutional Neural Network (CNN) genre classification models using images representing mel-spectrograms as input, as we want to explore the visualization of semantic descriptors in the input. Additionally CNNs are often represented in the state-of-the-art of genre classification.[CKN⁺19, GSS20, SPM⁺22] Furthermore, the second part of this thesis aims to investigate the extent to which semantic descriptors can be assigned to feature maps generated by the convolutional layers of CNN-based genre classification models. This approach aims to reveal which feature maps cover which semantic information and thus can enhance the introspective understanding of a model.

Research question 1: *To what extent can semantic descriptors be learned from audio inputs as used for a state-of-the art CNN genre classification model?*

For the evaluation of research question 1, it is needed to train the state-of-the-art models, selected based on a systematic literature review, for spectrograms as input and semantic descriptors as output. Learning descriptive labels is a multi-labelling problem. Therefore, frequently used multi-label performance metrics, namely ROC-AUC, micro-/macro-averaged precision and recall, are used for evaluation.

Research question 2 focuses on the examination on how accurately the explanations fit the given genre, intended to assess if predicted descriptors can serve as an explanation for a genre classification.

Research question 2: *To what extent are the explanations in form of semantic descriptors, provided by the explanation system, relevant to the ground truth genre?*

For the evaluation, a mapping between semantic descriptors and genres is needed to show the relevance of the predicted explanations to the given genre. Thus, an appropriate classification model needs to be selected, trained on a training set consisting of the

descriptive labels as input and the genre as outcome, and evaluated on a test set.

Research question 3: *To what extent can semantic descriptors be assigned to feature maps of a state-of-the-art CNN genre classification model?*

Research Question 3 aims to determine which semantic descriptors are best represented by which feature map of a CNN genre classification model. To assign semantic descriptors to feature maps, we train models for feature maps as input and descriptors as outcome. In this step we assume that the feature map for which predicting a semantic descriptor achieves the best performance, also contains semantic information that covers the semantic descriptor best.

1.3 Thesis Structure

The explanation system approach for state of the art genre classification models, the approach to assign semantic descriptors to feature maps generated by the convolutional layers of such models, as well as the dataset we use are introduced in Chapter 2. In Chapter 3 we determine and describe the state of the art genre classification model for which we apply the proposed approaches. A systematic literature review is conducted in Chapter 4 to provide an overview of the state of the art in learning semantic descriptors. The model performing best for learning descriptors for our explanation system is determined, its predictions are visualized, and the whole workflow of the explanation system is described in Chapter 5. In Chapter 6 we apply the second proposed approach, assigning semantic descriptors to feature maps, on the dataset selected in Chapter 2. Finally, we draw the conclusions of our work with respect to our research question and provide an outlook for future work in Chapter 7.

1.4 Related Work

Already in the beginning of artificial intelligence (AI), researchers have recognized that intelligent systems should explain their results [XUD⁺19]. Therefore, explainability is not a new topic and articles were published more than 40 years ago already covering the topic of explainability [SCDS77, Swa81]. Just as rule-based expert systems, with rules defined by human experts, can explain the reason for a negative or positive decision, so can a decision tree and is therefore a good example for an explainable structure of AI [XUD⁺19]. With the development of modern DNNs, however, the field of research for explainable AI has been revisited, since modern DNNs are not natively explainable by either the network itself or the developer of the network [XUD⁺19].

To explain decisions of deep learning models, that do not inherently have an explainable structure, we can distinguish between justification and introspective explanations. Justifications provide information, based on which the user can evaluate whether the decision was a good one or not [BC17]. However, justifications do not explain how

this decision was reached. On the contrary, introspective explanations aim to use the network itself to explain how a decision was made [BC17]. An example for introspective explanations is layer-wise relevance propagation (LRP) [MBL⁺19], which is a method to gain transparency for the decision of a deep neural network, by propagating the output relevance back through the network layers and assigning a relevance score to each input variable [XUD⁺19].

1.5 Convolutional Neural Networks (CNN)

Convolutional Neural Networks are a class of deep learning algorithms which have become very popular in recent years because of their performance in a variety of tasks such as image classification, object recognition, and segmentation. The network has its name from the fact that it uses a convolution, a mathematical function, to summarize local patterns by scanning the input and generating feature maps based on the used filters. [LBH15, GBC16, KSH17]

In the following, the core components of CNNs are briefly introduced and described.

1.5.1 Convolutional Layer

The main part of a Convolutional Neural Network is the convolutional layer, which contains a set of filters. The filters are set with different weights and are used to convolve the input by sliding over it and calculating the scalar-product between the elements of the filter and the input [MW21]. This process results in a set of feature maps that capture different aspects of the input image. Therefore the output of the convolutional layer is three dimensional with the third dimension being the number of feature maps, called channels. Usually, CNN consist of multiple convolutional layers, where each layer builds on the features learned by the previous layer to produce a more abstract representation of the input.[MW21] The number of filters, the size of the filter and the size of the strides, the step size with which the filter scans the input, are essential parameters of a convolutional layer. For the output of a convolution layer, usually a non-linear function is used as activation function. The most commonly used function is the rectified linear unit (ReLU), defined as $f(x) = \max(x, 0)$, where x is an element of the scalar-product between filter and input [YNDT18].

1.5.2 Pooling Layer

The pooling layer intends to down sample the feature maps after a convolutional layer and does not include any trainable parameters. The layer can be configured like a convolutional layer by setting the filter size and the stride. The max-pooling layer is the most commonly used and outputs the maximum value within the filter window and discards all other values [YNDT18]. Global-averaged-pooling is a pooling layer where for each feature map the average of all elements is calculated.

1.5.3 Fully Connected Layer

In the fully connected layer (or dense layer) each input neuron is connected to all output neurons and is used to create the final output. To connect the fully connected layer to the convolutional layers, typically a flatten layer, which resamples all feature maps to a one dimensional array, or global-average-pooling is used [YNDT18]. The activation function for the final output has to be set according to the task the network should fulfill. For a multi-class classification, softmax activation is used, which normalizes the output values so that the sum of all output values equals to one and thus the output values represent the class probabilities [YNDT18]. For a probability distribution, needed for binary classification and multi-label classification, a logistic sigmoid function is used as activation function, as it maps the values to a range of zero to one [GBC16].

1.5.4 Dropout

Dropout is a mechanism to prevent overfitting in neural networks. In each training epoch, a set of nodes is selected that should not be present in the training and their output is set to zero [Iza22].

1.6 Signal Processing

To use an audio file as an input of a CNN, the audio signal has to be processed to an multi-dimensional representation of the audio signal. The mel-spectrogram is one of the most commonly used representations of audio sequences used as input for CNNs. It divides the audio signal into frames using a window function and applies a fast fourier transformation (FFT) to each window to transform the time domain into the frequency domain, by providing the magnitudes of the spectral components. The magnitudes are then grouped into bins according to the mel scale, which maps frequencies to the approximate mel scale perceived by the human ear. An examples of a mel-spectrogram is shown in Figure 3.1 in chapter 3.

Explanation Approaches

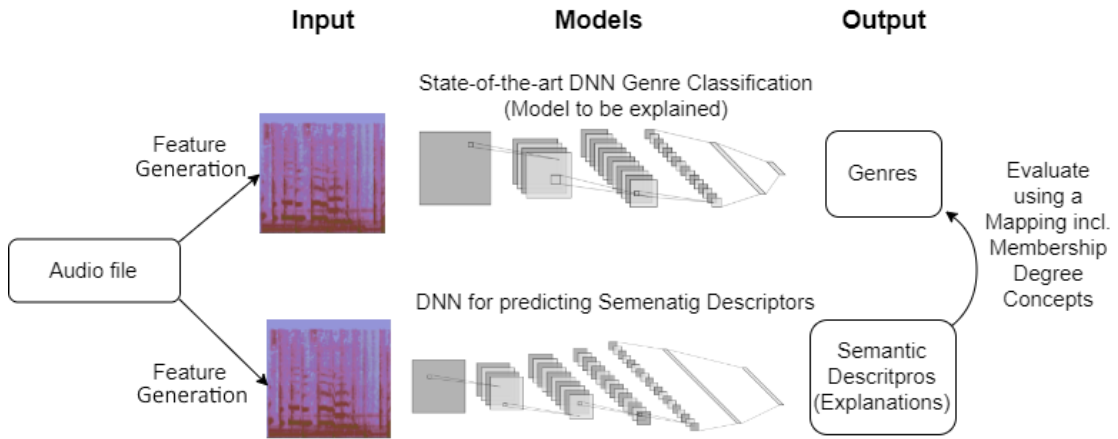
In this thesis we consider two different approaches to explain decisions of state-of-the-art genre classification models. The first approach, the explanation system, justifies the decisions by providing semantic descriptors. The second approach connects the feature maps of the convolutional layers of the state-of-the-art genre classification models and the semantic descriptors. In this chapter, these two proposed approaches are introduced.

2.1 Explanation System Approach with Semantic Descriptors

The main goal of the explanation system approach with semantic descriptors is learning music characteristics or semantic representations of music which should justify music genre-classification decisions and gain trust in the model to be explained. To develop such a explanation system, two main components are needed.

The first component is the training of a model to predict semantic descriptors, such as occurring instruments, timbre and mood, which should explain, justify or at least, give an indication for the genre decision of the model to be explained. Since a song has several characteristics, this is a multi-labelling problem. To evaluate models for learning semantic descriptors we use accuracy, micro-/macro-averaged precision and recall. To evaluate the relevance of the learned descriptors to the predicted genre and therefore to what extent the descriptors describe the genre-classification decision itself, we need something capable of telling how much certain characteristics fit certain genres. Since there does not exist a clear definition of what music characteristics a genre consists of, we create a mapping between all semantic descriptions of songs and their respective music genre. This mapping is the second component for the explanation system approach. The mapping has to include a membership degree to show how relevant the predicted

Figure 2.1: Schematic description of the relationship between individual components of the explanation system approach



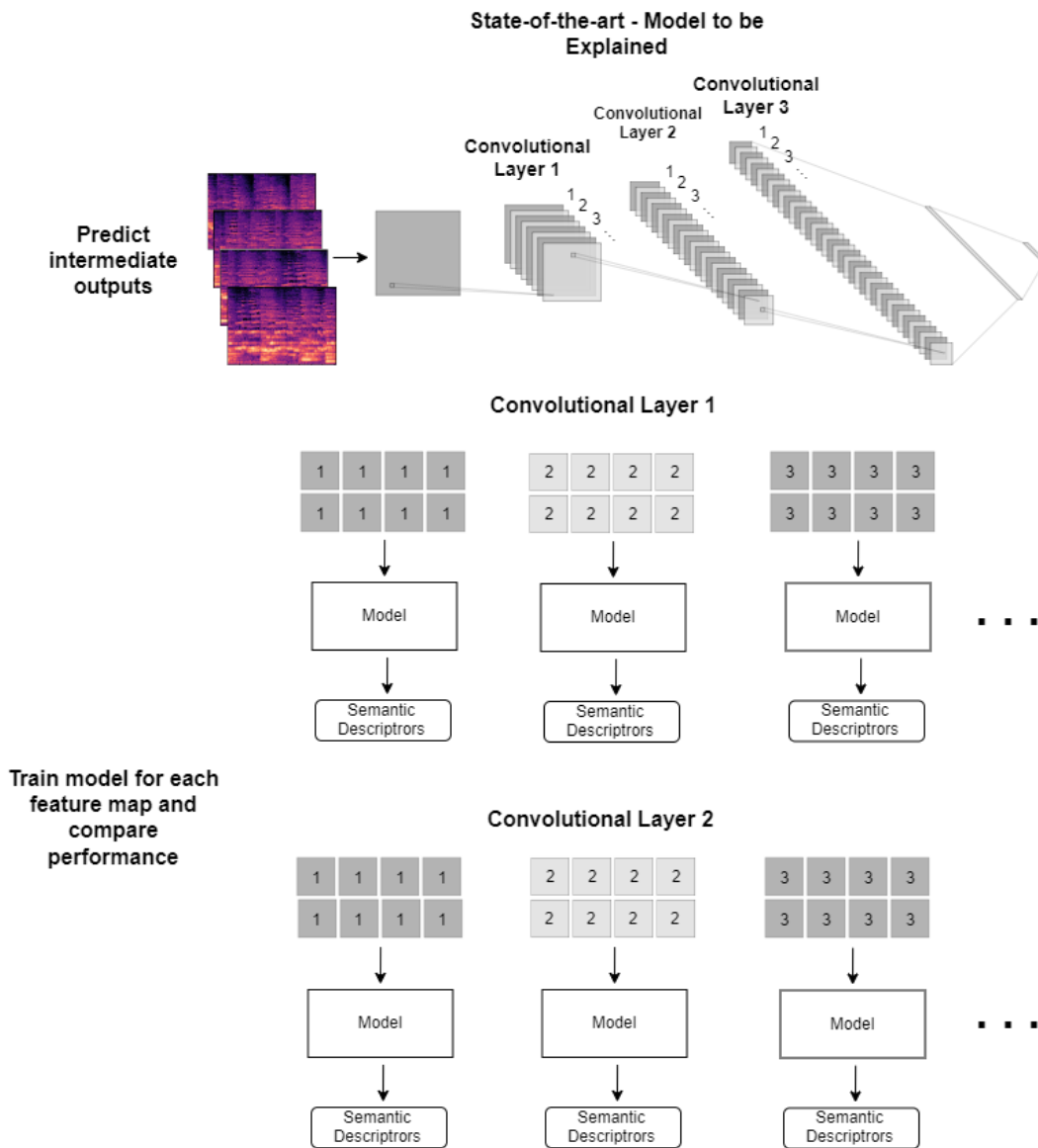
explanation is to the given genre. An overview of the explanation system approach is shown in Figure 2.1

In a further step, the found music properties are located in the spectrogram input of the descriptor learning model with Deep Taylor Decomposition, a variant of layer-wise relevance propagation [BBM⁺15], in order to improve the trustworthiness of explanation systems [HHDA18].

2.2 Assigning Descriptive Labels to Feature Maps generated by Convolutional Layers

Convolutional Neural Networks generate feature maps after each convolutional layer by applying filters to an input. The feature maps of the first convolutional layer are the result of applying one filter to the input spectrogram and extract low-level features [MW21]. The subsequent layers combine the features of the previous layer and thus extract more and more detailed information. In the second part of this thesis we want to assign feature maps to specific music characteristics. Therefore, we store the intermediate outcomes of the model to be explained for a set of audio files and train a model for each single feature map as input and music characteristics as output. We investigate possible differences in performance for different layers as well as for different feature maps within one layer. For this step, we consider Random Forest (RF), K-Nearest-Neighbour (k-NN) and Support Vector Machines (SVM) as models for training and testing the music characteristics for each feature map. It is to assume that we can obtain better performance for feature maps and specific music characteristics if a feature map has abstracted semantic information which is equivalent to a specific characteristic. This approach aims to determine which

Figure 2.2: Overview of Assigning Descriptive Labels to Feature Maps Approach



feature maps represent which music characteristics best or worst and, unlike the first approach, is not intended to explain individual music genres decisions but to facilitate a better understanding of the model to be explained.

2.3 Datasets

In order to implement and evaluate the two proposed approaches, a dataset that contains audio, for mel-spectrogram creation, genres and descriptive audio characteristics is needed. In the following we present frequently used datasets in the area of music information retrieval (MIR) research.

One can roughly divide such datasets into those created by annotating audio file with the goal of assigning characteristics and those created by collecting information from platforms that allow users to tag songs. The latter are therefore called social tags. This type of data collection is less costly and results in much larger datasets, however, also leads to much noisier data.

2.3.1 Annotated Semantic Information

CAL500

The Computer Audition Lab 500 (CAL500) dataset [TBTL08] contains 500 unique songs annotated with 135 music relevant properties of six semantic categories: genre, instruments, vocal characteristics, emotions, acoustic characteristics like tempo, energy and sound quality, and usage terms like in what situation this music is suitable. To annotate the audio files, 66 students were hired to listen to at least 30 seconds of each song before annotating it with the given set of the vocabulary. The 135 annotated music properties are mapped to 237 music tags by mapping all bipolar properties to two individual tags. For example, the characteristic *Energy Level*, which is tagged with 1 to 5, is mapped to *Low Energy* and *High Energy*. Afterwards the tags are reduced to 174 characteristics that are represented by at least five songs. For each of the 500 songs the creators provide Mel-frequency cepstral coefficients (MFCCs) but no audio files of the songs itself. MFCCs are audio representations like mel-spectrogram where the calculation is based on furier transformation and discrete cosine transform. For the CAL500, about 5,200 39-dimensional MFCCs are created per minute of audio content. By randomly subsampling, they select 10,000 MFCCs per song to obtain equally sized representation for each song.

CAL10K

The Computer Audition Lab 10K (CAL10K) dataset [TKT10] contains 10,870 songs of 4,597 different artists annotated with 153 genres (18 genres and 135 sub-genres) and 475 acoustic tags including instruments, vocal characteristics, emotions, acoustic characteristics and usage terms like in the CAL500. The annotations were obtained from Pandora's Music Genome Project and include between two and 25 tags per song. According to [TKT10], the tag selection and annotation were carried out by trained music experts as part of Pandora's Music Genome Project and can be considered acoustically objective. Like in CAL500, MFCCs are provided as audio representations but no audio

files itself. For each song of CAL10K six 5-second snippets, evenly distributed throughout the song, are selected and about 2,700 36-dimensional MFCCs are created.

MagnaTagATune

The MagnaTagATune dataset [LWM⁺09] consists of 25,863 audio files each with a length of 29 seconds, annotated with 188 music characteristics in total including 37 genre related tags and 151 other tags covering instruments, vocal characteristics, emotions and acoustic characteristics like tempo and energy. The audio files are provided as Mp3 files. The dataset was collected using the game TagATune [LVADC07], which is designed to gather music labels for songs from users.

Free Music Archive - FMA

The Free Music Archive (FMA) [DBVB17] is another dataset frequently used in music information retrieval and contains 106,574 audio tracks labeled with 161 genres. The dataset also contains artist metadata, like name, member, location and active years, and album metadata, like album title and release date. Furthermore the Echonest ID as well as audio features provided by Echonest¹ (now Spotify²) like a rate of acoustiness, danceability, energy and instrumentality, and tempo in bpm are provided for the FMA Dataset. Unfortunately, it does not contain any music characteristics other than genre and is therefore not suitable for the purpose of this work.

2.3.2 Social Tags

MSD - LastFM

The Million Song Dataset (MSD) [BMEWL11] provides audio features and metadata of a million popular music tracks. The team of MSD also collaborated with Last.fm and collected song-level tags and song similarity for 943,347 tracks of the MSD with the Last.fm API and created its own dataset, namely the Last.fm Dataset³. 505,216 tracks are tagged with at least one of the 522,366 tags. The tags made available by the API were created by users of Last.fm who can listen to music tracks and tag them independently, which, did not lead exclusively to audio-descriptive tags. However, the 522,366 tags covers music characteristics of the categories instruments, vocal characteristics, emotions, acoustic characteristics, usage terms, decades, countries and languages but also many user related tags like *favorites*, *nostalgic* and *i like*, and many incomprehensible tags like *aaa*.

Table 2.1: Comparison of the data sets regarding properties relevant for the master thesis

Dataset	Music Characteristics / Semantic Descriptors	None-Genre Tags	Genre Tags	Audio Files
Cal500	instruments, vocal characteristics, emotions, acoustic characteristics, usage terms	174	yes	no
Cal10K	instruments, vocal characteristics, emotions, acoustic characteristics, usage terms	475	yes	no
MagnaTag-ATune	instruments, vocal characteristics, emotions, acoustic characteristics	151	yes	yes
FMA	-	0	yes	yes
MSD - Last.fm	instruments, vocal characteristics, emotions, acoustic characteristics, usage terms, decades, countries, languages	> 500,000	yes	yes

2.3.3 Selection

The properties of the presented datasets relevant to our proposed approaches are summarized in Table 2.1. As Cal500 and Cal10K are not providing audio files and not even a mel-spectrogram we can not consider them for our approaches. FMA offers audio files but no music characteristics except for genres and is therefore also not suitable for the implementation of our approaches. The MSD-Last.FM dataset contains all the information we need, but is subject to a lot of cleanup due to the fact that it is a social tag dataset. Since MagnaTagATune also contains all the information we need and in addition it is an annotated dataset, which in contrast to CAL500 and CAL10K, also includes the audio files, we choose it to apply and evaluate our proposed approaches, where the contained property serves us as a semantic descriptor. Moreover, the MagnaTagATune dataset is used for all models we consider for the explanation system in their original papers, determined by the semantic literature research presented in chapter 4.

¹<https://web.archive.org/web/20170519050040/http://the.echonest.com/>

²<https://open.spotify.com/>

³<http://millionsongdataset.com/lastfm/>

Model to be Explained

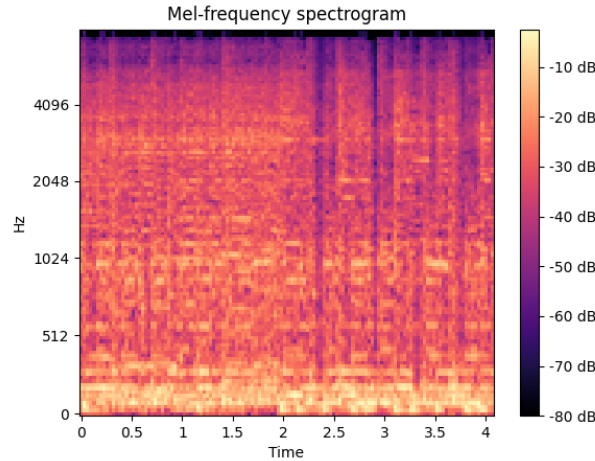
Since the goal of this thesis is to generate explanations and examine feature maps of CNN based music genre classification models, the first step is to specify a model to be explained on which we can apply and validate our explanation approaches.

Using Convolutional Neural Networks is a common way to perform music genre classification tasks on spectrograms and achieve good performance across the board [CKN⁺19, GSS20, SPM⁺22]. We implement the CNNs introduced in [CKN⁺19] and [PG20] which are reported as well-performing and also clearly describe the architectures as well as the implementation details of the networks, and select the better performing one as model to be explained. We pre-process the MagnaTagAtune dataset to obtain a dataset that can be used for training the genre classification models. Due to the nature of Deep Neural Networks, a large number of training data is needed to obtain a stable model. Thus, we implement the networks for an input of 128x128x1 like in [PG20] and cut the mel-spectrogram we obtain from a full audio file into appropriately sized parts in order to generate more training data.

3.1 Data Pre-processing

In the used dataset MagnaTagATune are 12 genre annotations within the top 50 annotations, namely, *classical*, *techno*, *electronic*, *rock*, *indian*, *opera*, *pop*, *classic*, *new age*, *dance*, *country* and *metal*. As the genres *classic*, *classical* and *opera* as well as *techno*, *electronic* and *dance* are often annotated together we combined them to the genres *classical* and *electronic* which finally leads to eight genres. After combining the genres we obtain 12,310 audio files annotated with a single genre and 13,550 annotated with multiple genres. The mel-spectrograms are created from the mp3 files using the open source Python library Librosa. We used the original sampling rate of the mp3 files of 16 kHz, a window size of 1024, a hop length of 512 and 128 mel bands. As the mel-spectrogram creation leads to a mel-spectrogram of the size 128x911, we can cut

Figure 3.1: Example of a 128x128 Mel-Spectrogram for the song 'Howl at the Moon' from Seismic Anamoly



13 mel-spectrograms for each audio file when using a hop size of 64. A representative example of such a mel-spectrogram is shown in Figure 3.1 for the song 'Howl at the Moon' from Seismic Anamoly which is labeled with the genre *rock*.

Of all audio files, which are only tagged with one genre, 295 are annotated with the genre *country*, as compared to 5,048 annotations with *classical* and 3,254 with *electronic*. After preserving 2,000 audio files for later testing purposes, only 242 are left annotated with *country*. Thus, we train the models once with fully balanced data where each genre is limited to 242 audio files, resulting in 3,146 mel-spectrograms each and once with data where each genre is limited to a maximum of 1,000 audio files to increase the number of training data and to determine whether it has an impact on the performance. On the one hand we can expect better performance for the semi-balanced data since we create more training data, but on the other hand Pelchat and Gelowitz [PG20] has observed the opposite. A listing of the precise number of calculated mel-spectrograms for each genre is shown in Table 3.1. The audio files are randomly split into training set (80%) and test set (20%) and only afterwards the mel-spectrograms are created, to avoid mel-spectrograms from the same audio file being present in both training and test set.

3.2 CNN Architecture following [CKN⁺19]

In this section we describe the architecture of the music genre classification model introduced in [CKN⁺19].

Table 3.1: Number of created Mel-Spectrograms for each Genre for Balanced and Semi-Balanced Data

No.	Genre Name	Balanced	Semi-Balanced
1	Rock	3,146	13,000
2	Indian	3,146	13,000
3	Pop	3,146	5,070
4	New Age	3,146	3,679
5	Country	3,146	3,146
6	Metal	3,146	5,902
7	Electronic	3,146	13,000
8	Classic	3,146	13,000
	Total	25,168	69,797

3.2.1 Convolutional Layers

The model consists of three convolutional blocks, composed of a convolutional layer, followed by a batch normalization and a max-pooling layer. For the convolutional layers, 64, 128 and 256 filters are used, each with a filter size of 3x3 and a step size of one. Rectifier Linear Unit (ReLU) are used as activation function. For the max-pooling layer, a pool size and strides of two are selected which leads to a halving of the resulting dimensions of each convolutional layer.

3.2.2 Fully Connected Layers

Subsequent to the three convolutional layers, a fully connected layer with 512 units and a subsequent dropout of 20% to prevent overfitting are used. To convert the output of the convolutional layers to one dimension, a flatten layer is used. Finally, there is a softmax layer to determine the probability of membership to each genre.

3.3 CNN Architecture following [PG20]

In this section we describe the architecture of the music genre classification model introduced in [PG20].

3.3.1 Convolutional Layers

This model contains six convolutional layers which are each followed by a max-pooling layer. The convolutional layers have a filter size of 2x2, step size of one, and use 64, 128, 256, 512, 1024 and 2048 as number of filters and Rectifier Linear Unit (ReLU) as the activation function. For the max-pooling layer, a pool size and strides of two are selected.

Table 3.2: Layer listing of Genre Classification Model following [CKN⁺19]

Layer	Output Shape	Param #
Input Layer	(128, 128, 1)	0
Convolutional Layer (3x3)	(128, 128, 64)	640
Batch Normalization	(128, 128, 64)	256
MaxPooling2D (2x2)	(64, 64, 64)	0
Convolutional Layer (3x3)	(64, 64, 128)	73,856
Batch Normalization	(64, 64, 128)	512
MaxPooling2D (2x2)	(32, 32, 128)	0
Convolutional Layer (3x3)	(32, 32, 256)	295,168
Batch Normalization	(32, 32, 256)	1,024
MaxPooling2D (2x2)	(16, 16, 256)	0
Flatten	(65536)	0
Dense	(512)	33,554,944
Dropout (0.2)	(512)	0
Dense (Softmax)	(8)	4,104
Total params:		33,930,504
Trainable params:		33,929,608
Non-trainable params:		896

3.3.2 Fully Connected Layers

The output of the convolutional layers is converted to one dimension by a flatten layer and is then passed to a fully connected layer with 4096 units with a subsequent dropout of 50%. The last layer is a softmax layer to determine the probability of membership to each genre.

3.4 Implementation Details

Both CNNs are implemented in Python using Keras from TensorFlow. The models are trained with a batch size of 64, 50 epochs and a validation split of 20%. Categorical cross-entropy is used as loss function and ADAM with the default learning rate of 0.001 as optimizer.

Table 3.3: Layer listing of Genre Classification Model following [PG20]

Layer	Output Shape	Param #
Input Layer	(128, 128, 1)	0
Convolutional Layer (2x2)	(128, 128, 64)	320
MaxPooling2D (2x2)	(64, 64, 64)	0
Convolutional Layer (2x2)	(64, 64, 128)	32,896
MaxPooling2D (2x2)	(32, 32, 128)	0
Convolutional Layer (2x2)	(32, 32, 256)	131,328
MaxPooling2D (2x2)	(16, 16, 256)	0
Convolutional Layer (2x2)	(16, 16, 512)	2,098,176
MaxPooling2D (2x2)	(8, 8, 64)	0
Convolutional Layer (2x2)	(8, 8, 1024)	8,390,656
MaxPooling2D (2x2)	(4, 4, 1024)	0
Convolutional Layer (2x2)	(4, 4, 2048)	33,558,528
MaxPooling2D (2x2)	(2, 2, 2048)	0
Flatten	(8,192)	0
Dense	(4,096)	33,558,528
Dropout (0.5)	(4,096)	0
Dense (Softmax)	(8)	32,776
Total params:		44,769,480
Trainable params:		44,769,480
Non-trainable params:		0

3.5 Result

For training the models with balanced data, we achieve an accuracy of 68.6% for the model following [CKN⁺19] and an accuracy of 63.1% for the model following [PG20], for the semi-balanced data we observe an accuracy of 77.1% and 75.5%, respectively. We observe that the validation loss fluctuates significantly after only a few epochs. Therefore, we introduce Keras' callback methods *ModelCheckpoint* and *LearningRateScheduler*. *ModelCheckpoint* saves the model weights at some frequency, to load the best weights after the training process. In our case, we save the weights for the best value of accuracy. The customizable *LearningRateScheduler* callback method is called before each epoch, gets the current epoch and learning rate as argument and returns the learning rate. We define the *LearningRateScheduler* to keep the initial learning rate for the first 10 epochs and decreases it exponentially after that. After applying these callback methods, the fluctuation of the validation loss is reduced (see Figure 3.2) and the accuracy of the

3. MODEL TO BE EXPLAINED

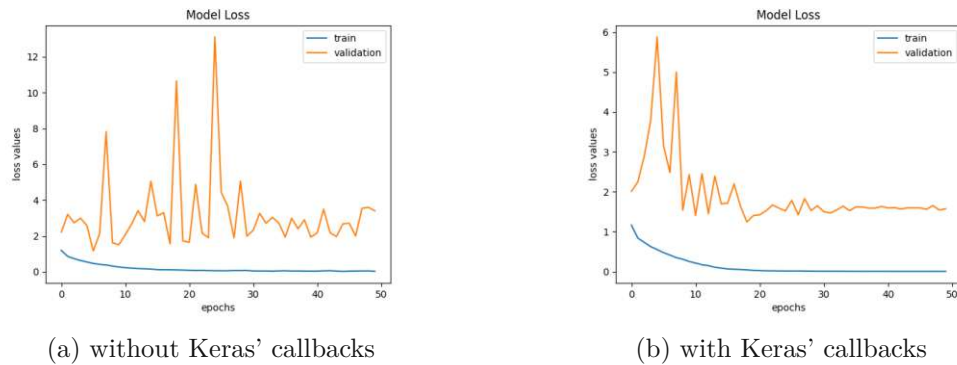


Figure 3.2: Validation loss fluctuation for CNN following [CKN⁺19] with and without using Keras' callback methods.

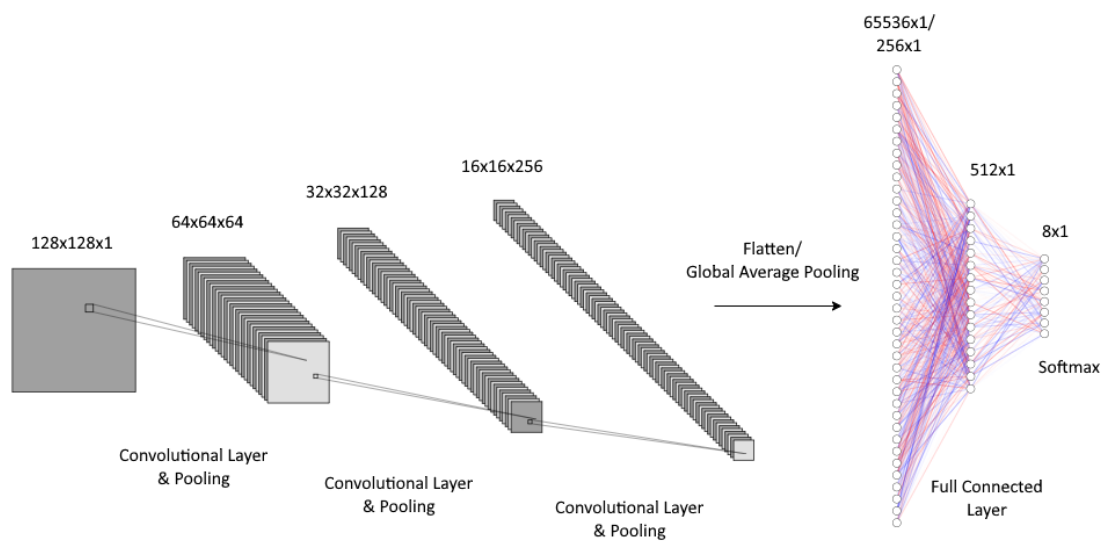
Table 3.4: Music Genre Classification CNNs - Performance

Model	Data	Accuracy	Precision	Recall	ROC-AUC
CNN [CKN ⁺ 19]	balanced Data	77.3%	77.9%	77.3%	95.2%
CNN [CKN ⁺ 19]	semi-balanced Data	79.8%	76.4%	72.1%	95.9%
CNN [PG20]	balanced Data	76.6%	77.0%	76.7%	94.8%
CNN [PG20]	semi-balanced Data	78.9%	74.7%	71.3%	95.3%

models improved. The models that are trained on semi-balanced data still outperforms the models trained on balanced data but the differences in performance is lower than before the callback methods were applied. For the CNN following [CKN⁺19], the accuracy for balanced data could be improved to 77.3% and the accuracy for semi-balanced data could be improved to 79.8%. For the CNN following [PG20], the accuracy for balanced data could be improved to 76.6% and the accuracy for semi-balanced data could be improved to 78.9%. The macro-averaged performances of the models after applying the callback methods are provided in Table 3.4. The original paper [CKN⁺19] could obtain an accuracy of 88.5% and the original paper [PG20] an accuracy of 85%, but are both trained on different datasets than the MagnaTagATune dataset.

The CNN following [CKN⁺19] trained on semi-balanced data achieves the best performance and is therefore selected as the model to be explained. Furthermore an overview of the selected model is shown in Figure 3.3.

Figure 3.3: Architecture of the selected CNN Genre Classification Model to be Explained (created with the online-tool NN-SVG [LeN19])



Learning Semantic Descriptors - Systematic Literature Review

In this section the results of a systematic literature review according to Kitchenham (2004) "Procedures for Performing Systematic Reviews" about learning descriptive labels from music files are presented. The aim of this systematic literature review is to provide an overview on state-of-the-art models in this field to be able to select a model for the explanation system approach introduced in this thesis.

4.1 Review Protocol

4.1.1 Rationale

This master thesis aims to learn explanatory labels from audio instances to use them as an explanation for predictions of black box models for genre classification. Since the development of an own model would be beyond the scope of the master thesis, a systematic literature review is carried out in order to be able to select a state-of-the-art model for the purpose of learning semantic descriptors.

4.1.2 Review Question

To what extent can descriptive labels be learned from music files?

The target audience are the user of the explanation system developed in the course of this master thesis. The subject matter is models for learning explanatory labels from audio files and the comparison of the models will be based on ROC-AUC. Predict descriptive labels for a song is a multi-label problem for which different approaches exist. The outcome is the performance and the approach used to solve the multi-label problem.

4.1.3 Search Strategy

Exclusion and Inclusion Criteria

The search is limited to models that use spectrograms as input, since the predicted semantic descriptors should be localized and visualized in the spectrogram in later steps of the master thesis to enhance confidence in the explanation system.

The models searched for, should be suitable for performing a multi-label problem for audio files, but the search is not restricted to any particular solutions for multi-label problems.

Resources

The searches are performed on the following databases:

- Scopus
- IEEE Xplore
- Web of Science

Search Terms

("music" OR "audio" OR "song" OR "audio-file") AND ("learning" OR "learn" OR "prediction" OR "predict" OR "model" OR "models" OR "cnn" OR "dnn" OR "neural network") AND ("multilabel" OR "multi-label") AND ("explanatory" OR "tangible" OR "descriptive" OR "semantic") AND ("labels" OR "tags" OR "representation" OR "representations" OR "descriptors" OR "explanation" OR "description" OR "explanations" OR "descriptions")
In IEEE Xplore and Web of Science the search term is applied to all fields. For Scopus the search is performed within title, abstract and keywords, as the search for using "All Fields" resulted in 3625 documents not related to the search term.

Search Filters

The filter is set to only include journals, magazines and conference proceedings published since 2019 in order to be limited to state-of-the-art models. Furthermore, the search is restricted to publications written in English.

Study Selection Procedures & Quality Assessment

Since this Systematic Literature Review is conducted in the course of this master thesis to ensure that the state-of-the-art tool used is carefully selected, the selection of the publications is performed only by the author himself, but to the best of his knowledge and belief. Only those articles are selected that can answer the review question and whose proposed model is reproducible and can be implemented in an acceptable amount of time.

4.2 Search

Conducting the automatic search to the respective search engines leads to the following number of found papers:

Scopus: 14

IEEE Xplore: 10

Web of Science: 15

The result of Scopus was actually 21 but with 7 duplicates. After eliminating the multiple appearances within the different search engines, the total number of papers found is 27.

Since we found off-topic publications by the author "Song", the search string is adapted to:

("music" OR "audio" OR "songs" OR "audio-file" OR "audio-files") AND ("learning" OR "learn" OR "prediction" OR "predict" OR "model" OR "models" OR "cnn" OR "dnn" OR "neural network") AND ("multilabel" OR "multi-label") AND ("explanatory" OR "tangible" OR "descriptive" OR "semantic") AND ("labels" OR "tags" OR "representation" OR "representations" OR "descriptors" OR "explanation" OR "description" OR "explanations" OR "descriptions")

The new result of the respective search engines are as followed:

Scopus: 14

IEEE Xplore: 4

Web of Science: 4

All these papers are analyzed based on their abstracts and filtered according to the inclusion and exclusion criteria. Only those papers that can be clearly excluded on the basis of their abstract are removed. Nevertheless, only the following 3 papers remain.

After a short manual search it became clear that the keyword "auto-tagging" is important for searching multi-label classification models for audio-files. Thus, another search term is determined to extend the list of found papers:

(„music“ OR „audio“ OR „songs“ OR „audio-file“ OR „audio-files“) AND („model“ OR „models“ OR „cnn“ OR „dnn“ OR „neural network“) AND („auto-tagging" OR "auto tagging" OR "auto-labeling" OR "auto labeling" OR "music tagging" OR "audio tagging") AND ("mel spectrogram" OR "mel-spectrogram") AND ("multi-label" OR "multilabel")

The result of the additional search with the respective search engines are as followed:

Table 4.1: Systematic Literature Review: Remaining papers after first iteration

Title	Authors	Year	Ref.
Evolutionary Approximation of Instrumental Texture in Polyphonic Audio Recordings	Igor Vatulkin	2020	[Vat20]
Multi-Label Sound Event Retrieval Using a Deep Learning-Based Siamese Structure with a Pairwise Presence Matrix	Jianyu Fan; Eric Nichols; Daniel Tompkins; Ana Elisa Méndez Méndez; Benjamin Elizalde; Philippe Pasquier	2020	[FNT ⁺ 20]
Zero-shot Learning for Audio-Based Music Classification and Tagging	Choi, J.; Lee, J.; Park, J.; Nam, J.	2020	[CN19]

Scopus: 5

IEEE Xplore: 33

Web of Science: 2

In total there are 38 individual papers and after the abstract-based pre-selection, only eight papers remain.

Most of the eliminated papers aims at a different model than CNN, do not describe a model at all or focus on a different, classification task like music genre classification. Additionally some papers focus on approaches taking raw wave or other formats than mel-spectrograms as input.

The complete list of publications which are analyzed in detail is shown in the following table.

4.3 Result

4.3.1 Summaries

1) "Evolutionary Approximation of Instrumental Texture in Polyphonic Audio Recordings" [Vat20] focuses on an algorithm to simultaneously approximate all onsets/chords from a given audio track and does not provide a model to learn semantic audio characteristics.

2) "Multi-Label Sound Event Retrieval Using a Deep Learning-Based Siamese Structure with a Pairwise Presence Matrix" [FNT⁺20] is about learning the similarity level between two audio instances using a Siamese Neural Network. It does not provide any model for learning descriptive labels of audio and is therefore not relevant for our review question.

Table 4.2: Systematic Literature Review: Remaining papers after elimination based on abstract

ID	Title	Authors	Year	Ref
1	Evolutionary Approximation of Instrumental Texture in Polyphonic Audio Recordings	Igor Vatulkin	2020	[Vat20]
2	Multi-Label Sound Event Retrieval Using a Deep Learning-Based Siamese Structure with a Pairwise Presence Matrix	Jianyu Fan; Eric Nichols; Daniel Tompkins; Ana Elisa Méndez Méndez; Benjamin Elizalde; Philippe Pasquier	2020	[FNT ⁺ 20]
3	Zero-shot Learning for Audio-Based Music Classification and Tagging	Choi, J.; Lee, J.; Park, J.; Nam, J.	2020	[CN19]
4	Implementation of Computationally Efficient and Accurate Music Auto Tagging	Rajendran, S.; Anandaraj, S.P.	2022	[RA22]
5	Music Auto-tagging Based on Attention Mechanism and Multi-label Classification	Ju, C., Han, L., Peng, G,	2022	[JHP22]
6	Loss Function Approaches for Multi-label Music Tagging	Knox, D; Greer, T; (...); Narayanan, S.	2021	[KGM ⁺ 21]
7	How Low Can You Go? Reducing Frequency and Time Resolution in Current CNN Architectures for Music Auto-tagging	Andres Ferraro; Dmitry Bogdanov; Xavier Serra Jay; Ho Jeon; Jason Yoon	2021	[FBJ ⁺ 21]
8	Towards an Efficient Deep Learning Model for Emotion and Theme Recognition in Music	Srividya Tirunellai Rajamani;Kumar Rajamani;Björn W. Schuller	2021	[RRS21]
9	Data-Driven Harmonic Filters for Audio Representation Learning	Minz Won;Sanghyuk Chun;Oriol Nieto;Xavier Serrc	2020	[WCNS20]
10	Improving Musical Tag Annotation with Stacking and Convolutional Neural Networks	Juliano Donini da Silva; Yandre Maldonado Gomes da Costa; Marcos Aurélio Domingues	2020	[dSdCD20]
11	Augmented Strategy for Polyphonic Sound Event Detection	Bolun Wang; Zhong-Hua Fu; Hao Wu	2019	[WFW19]

3) "Zero-Shot Learning for Audio-Based Music Classification and Tagging" [CN19] is investigating the zero-shot learning approach for audio domain. The experimental setting of this paper is about learning genres in an unsupervised way using instrumental annotations of the FMA dataset as side information in the first experiment and the LastFM tag annotations for Million Song Dataset in another experiment. The paper does not include a model related to learning descriptive labels. However, zero-shot learning is a topic with a lot of potential but the additional study of their possible contribution to explainability would go beyond the scope of this master thesis.

4) "Implementation of Computationally Efficient and Accurate Music Auto Tagging" [RA22] looks at the EfficientNet [TL19] model family. They evaluate the contained models against the MagnaTagATune dataset using mel-spectrograms of three second long segments and compared by means of accuracy, precision, recall, F1-score and ROC-AUC. Unfortunately the model is trained and assessed only for genres, which is why the results are not meaningful enough for our purposes.

5) "Music Auto-tagging Based on Attention Mechanism and Multi-label Classification" [JHP22] proposes a model for music auto-tagging based on CNN combined with attention mechanisms. In deep learning models, attention mechanisms aims to reinforce useful information and suppress the noise of useless information during the feature extraction [CCJM21]. The proposed model as well as the audio pre-processing are described in detail and the attention mechanism is reported as Squeeze-and-Excitation (SE) [HSS18]. The model is able to learn music characteristics and is therefore relevant for our thesis. The model achieves a ROC-AUC of 91.6% for the top 50 tags of MagnaTagATune dataset.

6) "Loss Function Approaches for Multi-label Music Tagging" [KGM⁺21] evaluates the best loss function for a short-chunk model proposed in "Evaluation of CNN-based Automatic Music Tagging Models" [WFBS20]. The paper describes the presented model in detail and additionally the code is provided ensuring full reproducibility. The model is compared to FCN and Musicnn (see next paragraph), a Self-Attention CNN, a CRNN, a raw waveform sample level CNN and a harmonic CNN (see paragraph 9) for three different datasets, including MagnaTagATune. The proposed model achieved the best performance on two datasets (MTAT, MSD) and shared first place with the harmonic CNN in the third dataset (MTG-Jamendo). The model is applicable for the explanation system approach and is therefore relevant for this thesis. ROC-AUC for top 50 tags of MagnaTagATune is reported with 91.3%.

7) "How Low Can You Go? Reducing Frequency and Time Resolution in Current CNN Architectures for Music Auto-tagging" [FBJ⁺21] investigates the impact of different settings for creating mel-spectrograms on performance in terms of fewer frequency bands and larger frame rates. This is done for a CNN from [CFS16] called FCN (Fully convolutional neural network) and the Musicnn [PPNCP⁺18] for the MagnaTagATune dataset as well as for the MillionSongDataset with LastFM tags for the top 50 tags respectively. The Musicnn outperforms the FCN with a ROC-AUC of 90.8% for MagnaTagATune. However, also FCN achieves a remarkable performance of ROC-AUC 89.4% for MagnaTagATune with only 4 convolutional layers [CFS16]. Both models are described in detail in their

original publication and are able to learn music characteristics, which make them applicable for the explanation system approach. Additionally, more details on Musicnn audio preprocessing and a git repository with the model implementation is provided by [FBJ⁺21].

8) "Towards an Efficient Deep Learning Model for Emotion and Theme Recognition in Music" [RRS21] proposes a CNN with a self-attention mechanism for emotion and theme recognition in music. The main goal of the investigation is the reduction of the floating point operations per second (FLOPS) to optimize the deployment of such models to facilitate the training on hardware with limited resources. The model itself as well as the implementation details and parameters such as learning rate, optimizer and epochs are described in detail. The model is trained and evaluated for emotion and theme data only and not for a dataset with general music properties, and is able to achieve a 72.6% ROC-AUC.

9) "Data-Driven Harmonic Filters for Audio Representation Learning" [WCNS20] introduced a method to create a new kind of input spectrograms for convolutional neural networks in the audio segment. As harmonic structure is an important factor in human auditory perception, their approach is to extend the common dimensions of spectrograms with harmonic as a third dimension by applying harmonic filters after the short-time Fourier Transformation (STFT). The CNN used to evaluate the proposed method is the same as the short chunk model described in [WFBS20].

10) "Improving Musical Tag Annotation with Stacking and Convolutional Neural Networks" [dSdCD20] presents two approaches of music auto-tagging with CNN combined with stacking. Stacking is a technique in which at least two stages are used and the outcome of a previous stage is used as input for, or at least affects, the next stage. In this paper both approaches used two stages with the same CNN. In the first approach the same input is used for both stages but in stage two the weights are initialized with the weights of stage one. In the second approach an autoencoder neural network, a CNN consisting of two phases, encode and decode, is used. The encode phase is intended to learn the desired result and is reducing the dimension of the input. The decode phase is to scale up the dimension and is used in stage one to create the input of stage two. For both proposed approaches the networks and the generation of the mel-spectrogram are described in-depth and are applied to the datasets FMA, MagnaTagATune and Million Song Dataset. The first approach achieves better performance as compared to the second approach and has a ROC-AUC of 89% for the MagnaTagATune dataset.

11) "Augmented Strategy for Polyphonic Sound Event Detection" [WFW19] describes a new method for augmenting audio events to increase the amount of training data for sound event detection. The augmentation strategy also includes a CNN for audio tagging which is described in the paper. However, the entire strategy, and thus the tagging, is applied to a dataset with annotated sound events in domestic environments and is therefore not meaningful for our objective.

4.3.2 Selection

We can select five models, each with a ROC-AUC of approximately 90% for the task of learning descriptive labels for music using the dataset MagnaTagATune. Since our goal is a model that learns descriptive labels to explain genre classifications, we will remove all genre related labels from the dataset before we train the models. Thus we do not train them for the same data as the original papers do, as no such restriction is made at any of the presented papers.

Table 4.3: Systematic Literature Review: Resulting models and their performance for the top 50 tags of MagnaTagATune

ID	Model Name	ROC-AUC
5	SE-CNN [JHP22]	91.6%
6	Short-Chunk CNN [WFBS20]	91.3%
7	FCN [CFS16]	89.4%
7	Musicnn [FBJ ⁺ 21]	90.8%
10	Stacked CNN [dSdCD20]	89.0%

Explaining Genre Classification with Semantic Descriptors

In this chapter an approach to explain convolutional neural network genre decisions based on descriptive labels is presented. For learning the descriptive labels for audio files we chose a model based on the results of the systematic literature review described in the previous chapter. Further, we train a model to map descriptive labels to genres to assess how accurately the predicted descriptors match the ground truth genre. The visualization of the generated explanations can enhance trustworthiness of the explanation system [HHDA18], therefore we visualize the descriptive label in the input spectrogram. As discussed in Section 2.3, we use the MagnaTagATune dataset to train and test our models.

5.1 Learning Semantic Descriptors

In this section, we determine a model for predicting semantic descriptors that will serve as the explanations for our explanation system approach. In the overview Figure 2.1, this part is shown in the bottom section. For this reason, we describe in detail all models found in the systematic literature review in terms of mel-spectrogram creation and model architecture, train them and compare their performance to be able to determine the most appropriate model for our application.

5.1.1 Data Pre-processing

The MagnaTagATune dataset has a total of 188 tags. Since several of them are synonyms or are very similar to others, we analyze them and determine tags as equivalent as shown in Table 5.1. Since we want to use this tag set to describe and explain genre decisions, genre-tags are excluded. Further, we exclude the tags *english* and *not english*. Thus, the excluded tags are *rock*, *indian*, *pop*, *new age*, *country*, *metal*, *electronic*, *classic*, *rap*, *hip*

5. EXPLAINING GENRE CLASSIFICATION WITH SEMANTIC DESCRIPTORS

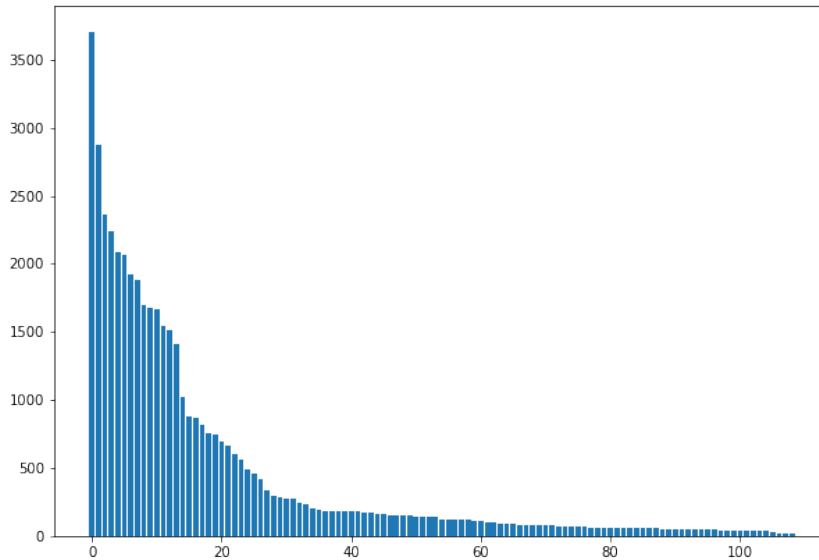
hop, reggae, punk, blues, not opera, not rock, not classical, funk, folk, jazz, disco, not english and *english*. Finally, we obtain a total of 108 remaining tags which are used as the descriptive labels for our explanation approach.

Table 5.1: Equivalent Tags of MagnaTagATune

Synonyms	
classic , classical, classical, opera, operatic, female opera, male opera	male vocals , male, male singer, male vocal, male voice, man, man singing, men
female vocals , female, female singer, female singing, female voice, woman, woman singing, women	no vocals , no singer, no singing, no vocal, no voice, no voices, instrumental
electronic , electro, electronica, electric, dance, techno	jazz , jazzy
beat , beats	guitar , guitars
chant , chanting	flute , flutes
choir , choral	harpsichord , harpsicord
drums , drum	vocals , vocal, voice, voices
fast , fast beat, quick	rock , soft rock, hard rock
metal , heavy metal	orchestra , orchestral
horn , horns	quiet , silence
indian , india	singer , singing
no drums , no beat	space , spacey
string , strings	violin , violins
synthesizer , synth	strange , weird

As most models do not take a whole audio file but smaller sections as input we keep 2,000 audio file aside, which are tagged with only one genre, to use them for testing the models on an audio file level as well as for the descriptor to genre mapping to asses how accurate the predicted descriptors of all sections fit the ground truth genres. From all remaining audio file, we filter out all those that do not have at least one tag, resulting in a total set of 18,373 audio files. An overview of how often the respective tags are tagged within the remaining instances can be seen in Figure 5.1. While the top ten tags are all tagged at least 1,925 times, the top 50 tags already only occur at least 173 times. We train all models for the top 50, top 80 and all 108 tags and compare them based on performance metrics but also with the descriptor-genre mapping model we provide in section 5.2.

Figure 5.1: Tag Histogram of the 108 remaining Tags after Pre-Processing the MagnaTagATune Dataset



5.1.2 SE-CNN

Music automatic tagging is the process of learning descriptive labels from audio files to be able to predict music characteristics for non-tagged audio files and can be divided into the tasks of audio pre-processing (mel-spectrogram creation), feature extraction (convolutional layers) and classification (fully connected layer) [JHP22]. Ju et al. [JHP22] focused on the improvement of feature extraction and proposed a CNN model with an attention mechanism. In conventional CNN models, during the convolution process, no relation is learned between the feature maps output by a convolution layer [JHP22]. The attention mechanism addresses this problem and helps the model to focus on important parts of input data by assigning weights to them. This way, the model can give more importance to relevant information while ignoring irrelevant data [JHP22]. To take such dependencies between feature maps into account, Ju et al. [JHP22] added the Squeeze-and-Excitation (SE) block [HSS18] to the model, which also gives the model its name.

The following paragraphs provide a detailed description of our implementation of the SE-CNN model as proposed in [JHP22].

Mel-Spectrogram Creation

Since the input shape dimension of the model is 128x628 and the audio data are resampled to 11,025 Hz, we conclude the authors use a hop length of 256 and the 128 mel bins for the creation of the mel-spectrogram. Since the length of the fast fourier transformation (FFT) window is not documented we use a length of 512. Additionally, we train the model for audio data with their original sampling rate of 16 kHz. As this results in a mel-spectrograms with a shape of 128x911, we cut them into 128x628 slices with a hop length of 90 to get four slices from each audio file.

Implementation Details

The model consists of 5 convolutional layers which are all followed by a Squeeze-and-Excitation (SE) block. The SE block consists of a global average pooling of the output of a convolutional layer, a fully connected layer with ReLU activation and a fully connected layer with sigmoid activation, which output is multiplied by the output of the respective convolutional layer. Max-pooling layers are added after the first three SE blocks with filter size 2x4 and after the fifth SE block with filter size 2x2. After the last max-pooling layer, a flatten layer, a batch normalization layer, a dropout layer with a rate of 0.6 and finally a sigmoid layer are added. The output dimension of the sigmoid layer is accordingly set to 50, 80 and 108, since we train it for the top 50, top 80 and all 108 tags as described in Section 5.1.1. A more detailed description of the used layers can be found in Table 5.2. For training the model, the ADAM optimizer with a learning rate of 0.0001 following the original paper is used.

5.1.3 Short-Chunk CNN

Won et al. [WFBS20] provide a simple seven-layer CNN for 3.69s audio excerpts. This model outperforms state-of-the-art music auto-tagging models for the top 50 tags of MagnaTagATune. Because of the use of such short audio snippets the model is called "Short-Chunk CNN".

The following paragraphs provide a detailed description of the Short-Chunk CNN model proposed in [WFBS20].

Mel-Spectrogram Creation

The mel-spectrograms are created for the audio origin sample rate of 16 kHz, a FFT window of length 512, a hop length of 256 and 128 mel bins, which leads to a mel-spectrogram of the size 128x1821 for each of 29 seconds long audio files. To obtain the reported 3.69-second excerpts, we cut the total mel-spectrogram into 230 long pieces with a hop length of 115, resulting in 14 mel spectrograms for each and a total of 257,222 for all 18,373 audio files.

Table 5.2: Detailed description of SE-CNN Model [JHP22] (for 108 tags)

Layer (Filter/Pool Size)	Output Shape	Nr. of Parameters
Input Layer	(128, 628, 1)	0
Convolutional Layer (ReLU) (3x7)	(128, 628, 50)	1,100
SE-Block	(128, 628, 50)	0
MaxPooling2D (2x4)	(64, 157, 50)	0
Convolutional Layer (ReLU) (3x5)	(64, 157, 100)	75,100
SE-Block	(64, 157, 100)	0
MaxPooling2D (2x4)	(32, 39, 100)	0
Convolutional Layer (ReLU) (3x3)	(32, 39, 70)	63,070
SE-Block	(32, 39, 70)	0
MaxPooling2D (2x2)	(16, 19, 70)	0
Convolutional Layer (ReLU) (3x3)	(16, 19, 70)	44,170
SE-Block	(16, 19, 70)	0
Convolutional Layer (ReLU) (3x3)	(16, 19, 70)	44,170
SE-Block	(16, 19, 70)	0
MaxPooling (2x2)	(8, 9, 70)	0
Flatten	(5040)	0
Batch Normalization	(5040)	20,160
Dropout (0.6)	(5040)	0
Dense (Sigmoid)	(108)	544,428
Total params:		793,738
Trainable params:		783,738
Non-trainable params:		10,080

Implementation Details

The model is composed of seven convolution layers with 3x3 filters, each followed by a batch normalization, a ReLU and a max-pooling layer with a pool size of 2x2. The first two convolutional layers use a number of 128 filters, the following four layers use 256 filters and the last layer use 512 filters, resulting in an output of 128, 256 and 512 feature maps, respectively. Subsequently, a flatten layer, a dense layer with 512 units and a batch normalization is added. Finally, another dense layer with 50 units followed by a dropout of 0.5, a ReLU layer and finally the sigmoid activation is added. For the training of the model, the ADAM optimizer with a learning rate of 0.0001 is used. A detailed listing of the layers can be found in Table 5.3. We adjusted the last four layers to an outcome of 80 and 108 depending on which tag set is used for training.

Table 5.3: Detailed description of Short-Chunk CNN Model [WFBS20] (for 108 tags)

Layer (Filter/Pool Size)	Output Shape	Nr. of Parameters
Input Layer	(128, 230, 1)	0
Convolutional Layer (3x3)	(128, 230, 128)	1,280
Batch Normalization	(128, 230, 128)	512
ReLU	(128, 230, 128)	0
MaxPooling2D (2x2)	(64, 115, 128)	0
Convolutional Layer (3x3)	(64, 115, 128)	147,584
Batch Normalization	(64, 115, 128)	512
ReLU	(64, 115, 128)	0
MaxPooling2D (2x2)	(32, 57, 128)	0
Convolutional Layer (3x3)	(32, 57, 128)	295,168
Batch Normalization	(32, 57, 128)	1,024
ReLU	(32, 57, 128)	0
MaxPooling2D (2x2)	(16, 28, 128)	0
Convolutional Layer (3x3)	(16, 28, 256)	590,080
Batch Normalization	(16, 28, 256)	1,024
ReLU	(16, 28, 256)	0
MaxPooling2D (2x2)	(8, 14, 256)	0
Convolutional Layer (3x3)	(8, 14, 256)	590,080
Batch Normalization	(8, 14, 256)	1,024
ReLU	(8, 14, 256)	0
MaxPooling2D (2x2)	(4, 7, 256)	0
Convolutional Layer (3x3)	(4, 7, 256)	590,080
Batch Normalization	(4, 7, 256)	1,024
ReLU	(4, 7, 256)	0
MaxPooling2D (2x2)	(2, 3, 256)	0
Convolutional Layer (3x3)	(2, 3, 512)	1,180,160
Batch Normalization	(2, 3, 512)	2,048
ReLU	(2, 3, 512)	0
MaxPooling2D (2x2)	(1, 1, 512)	0
Flatten	(512)	0
Dense	(512)	262,656
Batch Normalization	(512)	2,048
Dense	(108)	55,404
Dropout (0.5)	(108)	0
ReLU	(108)	0
Sigmoid	(108)	0
Total params:		3,721,708
Trainable params:		3,717,100
Non-trainable params:		4,608

5.1.4 FCN-4

Choi et al.[CFS16] evaluated fully convolutional neural networks for automatic music tagging with a varying number of convolutional layers and different inputs. The 4-layer architecture (named FNC-4) using mel-spectrograms as an input, achieve state-of-the-art performance and outperform the other proposed models for the top 50 tags of MagnaTagATune. As Ferraro et al.[FBJ⁺21] provide an adjustment for the FCN-4 model for smaller inputs, we consider this implementation as well. The adaption of FCN-4 is called VGG-CNN as it is based on a Convolutional Neural Network from the Visual Geometry Group (VGG) [SZ15] which develops models for the computer vision field.

In the following paragraphs we describe the implementation of the FCN-4 model as proposed in [CFS16] and the implementation of FCN-4 adaptation as described in [FBJ⁺21].

Mel-Spectrogram Creation

For the mel-spectrogram creation the audio is resampled to 12 kHz and FFT with a window and hop length of 256 is applied. Because 96 mel bins are used, the result for each 29 second audio file is a 96x1366 mel-spectrogram, which is used as input for the model.

Implementation Details

The model consists of four convolutional layers, each with a filter size of 3x3 and a number of filters of 128, 384, 768, and 2048, respectively. Subsequent to each convolutional layer, batch normalization, ReLU activation and dropout of 0.5 is added. The output layer is a dense layer with sigmoid activation and 50 units. Additionally, we also adjust the output layer to 80 and 108 units according to the tag set used for training. A detailed listing of the layers can be found in Table 5.4. For training the model, the ADAM optimizer with a learning rate of 0.0001 is used.

VGG-CNN (FCN-4 Adaptation)

Ferraro et al. [FBJ⁺21] presents max-pooling layer adjustments for smaller mel-spectrogram input sizes for the FCN-4 and call it VGG-CNN. Furthermore, Won et al. [WFBS20] shows that mel-spectrogram from shorter audio excerpts perform better at music auto-tagging. Therefore, instead of reducing the input size by creating a smaller mel-spectrogram with different FFT window sizes and hop lengths, we create a mel-spectrogram with an FFT window of length 512 and a hop length of 256, resulting in a mel-spectrogram of size 128x1821, and split it into pieces of size 128x342 with a hop size of 171. This setting results in nine mel-spectrograms per audio file, increasing the amount of training data from 18,373 for FCN-4, as it uses only one mel-spectrogram per audio file, to 165,357 for VGG-CNN. According to [FBJ⁺21] we adjusted the pool size of the third max-pooling layer from 3x8 to 4x8 and the stride, the step size of the pooling window, from 2x4 to

5. EXPLAINING GENRE CLASSIFICATION WITH SEMANTIC DESCRIPTORS

Table 5.4: Detailed description of FCN-4 Model [CFS16] and the adaption based on [FBJ⁺21] (for 108 tags)

Layer	Setting		Output Shape		Nr. of Parameters
	Origin	Adaptation	Origin	Adaptation	
Input Layer			96x1366x1	128x342x1	0
Convolutional Layer	Filter:3x3	Filter:3x3	96x1366x128	128x342x128	1,280
Batch Normalization			96x1366x128	128x342x128	512
ReLU			96x1366x128	64x86x128	0
MaxPooling2D			48x342x128	64x86x128	0
Dropout	0.5	0.5	48x342x128	64x86x128	0
Convolutional Layer	Filter:3x3	Filter:3x3	48x342x384	64x86x384	442,752
Batch Normalization			48x342x384	64x86x384	1,536
ReLU			48x342x384	64x86x384	0
MaxPooling2D			24x86x384	32x22x384	0
Dropout	0.5	0.5	24x86x384	32x22x384	0
Convolutional Layer	Filter:3x3	Filter:3x3	24x86x768	32x22x768	2,654,976
Batch Normalization			24x86x768	32x22x768	3,072
ReLU			24x86x768	32x22x76	0
MaxPooling2D			12x22x768	11x6x768	0
Dropout	0.5	0.5	12x22x768	11x6x768	0
Convolutional Layer	Filter:3x3	Filter:3x3	4x8x2048	4x2x2048	14,157,824
Batch Normalization			4x8x2048	4x2x2048	8,192
ReLU			4x8x2048	4x2x2048	0
MaxPooling2D			1x1x2048	1x1x2048	0
Dropout	0.5	0.5	1x1x2048	1x1x2048	0
Dense			108	108	221,292
Sigmoid			108	108	0
Total params:					17,491,436
Trainable params:					17,484,780
Non-trainable params:					6,656

3x4 and the pool size of the fourth max-pooling layer from 4x8 to 4x2. A detailed listing of the CNN layers and their settings can be found in Table 5.4. For training the model, the ADAM optimizer with a learning rate of 0.001 is used.

5.1.5 Musicnn

Ferraro et al. investigated the performance for different settings in mel-spectrogram creation for the Musicnn model introduced in [PPNCP⁺18] and adjusts the filter sizes of the CNN layers for different input sizes in such a way that the output dimensions of the last convolution layer keep the same for all input sizes. In addition, the size of the audio segments used is set to three seconds, as they achieve better performance than with the original 15-second segments [FBJ⁺21].

Mel-Spectrogram Creation

The mel-spectrogram is created with a sample rate of 16 kHz, a FFT with a window size of 512 and hop length of 256 and 128 mel bins, resulting in a mel-spectrogram of size 128x1821. To obtain the three second segments the mel-spectrogram is divided into 128x188 sized mel-spectrograms leading to 18 mel-spectrogram per audio file and 330,714 mel-spectrograms in total.

Implementation Details

The architecture of the Musicnn consists of two different sets of convolutional layers applied in parallel on the input mel-spectrogram. One set uses the filter shapes 1x165, 1x128, 1x64 and 1x32, with the aim of learning temporal patterns. Each of these convolutional layers uses ReLU activation and a subsequent batch normalization. The second set of convolutional layers uses the filter shapes 1x51, 3x51, 7x51, 1x115, 3x115, and 7x115, the ReLU activation and are followed by batch normalization and a max-pooling layer. The aim of the second convolutional layer set is to learn timbre related features. The results of the parallel applied convolutional layers are concatenated and further processed in three additional convolutional layers with ReLU activation, two pooling layers (one max-pooling and one global pooling) and a dense layer with 200 units and ReLU activation. Finally, a dense layer of 50, 80 or 108 units depending on the number of tags we train the model with and sigmoid activation is added as output layer. ADAM with a learning rate of 0.001 is used as optimization method. For a more detailed information of the layers we refer to the origin publication of Musicnn [PPNCP⁺18] or the git repository¹ of Ferraro et al.

5.1.6 Stacked CNN

Donini da Silva et al. have proposed two different approaches for music auto-tagging with mel-spectrograms using CNN architectures with stacking technique, which means using at least two training stages, while the prediction generated after the first training are used as input for the second training of the CNN [dSdCD20]. One approach is based on an auto-encoder, a CNN architecture which first compress an image to a smaller dimensioned representation (encoder) and then reconstructs the compressed data to

¹<https://github.com/andrebola/EUSIPCO2020>

an image similar to the original (decoder). In the auto-encoder approach, first the whole auto-encoder is used to produce mel-spectrograms approximately 50% smaller than the original one. Afterwards only the encoder is used to predict tags taking the previous generated mel-spectrograms as input. The second architecture consists only of convolutional layers, batch normalization, dropout, max-pooling and dense layers, and uses the same input for both training stages, except that in stage one the weights are randomly initialized and in stage two the learned weights from stage one are used [dSdCD20].

Even though the performance of the second model hardly improve in stage two, it achieves better performance with 89% ROC-AUC for the MagnaTagATune Dataset than the first model with ROC-AUC of 81%, and therefore is implemented and evaluated for the purpose of this work.

The following two paragraphs present details on the implementation of the model based on the original paper [dSdCD20] and the repository².

Mel-Spectrogram Creation

For the mel-spectrogram creation the audio is resampled to 22,050 Hz, FFT is applied with a window size of 8,192 and hop length of 2,048 and 128 mel bins are used. This configuration results in 128x314 sized mel-spectrograms.

Implementation Details

The selected stacked CNN model has seven convolutional blocks, each block consisting of a convolutional layer with a 3x3 filter and a ReLU activation, a batch normalization layer, a dropout layer and a max-pooling layer with a pool size of 2x2. The number of output filters of the convolutional layers are 16, 32 and 64, respectively, for the first three convolutional layers, 128 for the fourth and fifth convolutional layers, and 256 for the last two convolutional layers. The rate of the dropout layers is 0.2 for the first two blocks, 0.25 for the third block and 0.3 for the last four blocks. Following the convolution blocks, there is a dense layer with 256 units, one with 128 units, and an output layer with sigmoid activation and a number of units equal to the number of tags to be learned. A detailed listing of the layers can be found in Table 5.5. For training the model, the ADAM optimizer with a learning rate of 0.001 is used.

5.1.7 Results

The 18,373 audio files are split into 80% training set and 20% test set. The training set is split again with a ratio of 9:1 for validation purposes. Afterwards, the mel-spectrograms are created according to the respective models. We train all models with 120 epochs but reduced the learning rate from epoch 80 using the *LearningRateScheduler* callback method of Keras. We also used Keras' callback method *EarlyStopping* to stop the training

²<https://gitlab.com/jdonini/stacking-audio-tagging>

Table 5.5: Detailed description of Stacked CNN Model [dSdCD20] (for 108 tags)

Layer (Filter/Pool Size)	Output Shape	Nr. of Parameters
Input Layer	(128, 314, 1)	0
Convolutional Layer (ReLU) (3x3)	(128, 314, 16)	160
Batch Normalization	(128, 314, 16)	64
Dropout (0.2)	(128, 314, 16)	0
MaxPooling2D (2x2)	(64, 157, 16)	0
Convolutional Layer (ReLU) (3x3)	(64, 157, 32)	4,640
Batch Normalization	(64, 157, 32)	128
Dropout (0.2)	(64, 157, 32)	0
MaxPooling2D (2x2)	(32, 78, 32)	0
Convolutional Layer (ReLU) (3x3)	(32, 78, 64)	18,496
Batch Normalization	(32, 78, 64)	256
Dropout (0.25)	(32, 78, 64)	0
MaxPooling2D (2x2)	(16, 39, 64)	0
Convolutional Layer (ReLU) (3x3)	(16, 39, 128)	73,856
Batch Normalization	(16, 39, 128)	512
Dropout (0.3)	(16, 39, 128)	0
MaxPooling2D (2x2)	(8, 19, 128)	0
Convolutional Layer (ReLU) (3x3)	(8, 19, 128)	147,584
Batch Normalization	(8, 19, 128)	512
Dropout (0.3)	(8, 19, 128)	0
MaxPooling2D (2x2)	(4, 9, 128)	0
Convolutional Layer (ReLU) (3x3)	(4, 9, 256)	295,168
Batch Normalization	(4, 9, 256)	1,024
Dropout (0.3)	(4, 9, 256)	0
MaxPooling2D (2x2)	(2, 4, 256)	0
Convolutional Layer (ReLU) (3x3)	(2, 4, 256)	590,080
Batch Normalization	(2, 4, 256)	1,024
Dropout (0.3)	(2, 4, 256)	0
MaxPooling2D (2x2)	(1, 2, 256)	0
Flatten	(512)	0
Dense	(256)	131,328
Dense	(128)	32,896
Dense	(108)	13,932
Sigmoid	(108)	0
Total params:		1,311,660
Trainable params:		1,309,900
Non-trainable params:		1,760

Table 5.6: SE-CNN - Performance

Top x Tags	ROC-AUC	Accuracy	Averaging	Precision	Recall
50	87.3%	95.5%	micro	63.2%	30.9%
			macro	37.0%	19.5%
80	85.7%	97.0%	micro	61.6%	29.6%
			macro	31.4%	13.3%
108	85.7%	97.6%	micro	61.1%	28.8%
			macro	24.7%	10.2%

process if the validation loss could not be improved within 30 epochs. After training, the best weights are loaded based on validation accuracy (binary) and the models are evaluated on the test set.

Since the finally selected model is applied to a whole audio file as part of the explanation system, but are trained for smaller mel-spectrogram sections, we also test the models on audio file level on the 2000 audio files set aside, and determined the performance on each individual tag. Therefore, we created a mel-spectrogram for each of the 2000 audio file and cut out as many mel-spectrogram frames as possible to create with a window size equal to the input length and a hop size half the length of the input frame for the respective model. We applied the models to each frame and collected all tags from all frames for each audio file. We have also conducted experiments where we have considered only tags that occur more frequently than certain threshold values, but achieve the best performance for collecting each tag. Afterwards, the overall performance as well as the performance on the different tags are determined.

For assessing the performances we focus on recall. The reason for this is the very sparse database, which drives accuracy upwards. A model that predicts no descriptor at all would have a very high accuracy for our dataset. Since songs of the MagnaTagATune dataset are tagged by non-experts in the context of a game, whose main goal was not to tag the music complete, there are also many music characteristics per song that are not tagged. This also means that we can rather tell that a missing prediction (false negative) is actually a missing characteristic than a falsely predicted descriptor (false positive) is actually not a characteristic of the song. Since precision considers the false positive rate in contrast to recall we focused recall, but did not completely disregard precision, since models that always predict all tags would result in 100% recall.

SE-CNN

For SE-CNN we achieve the best results for a batch size of 32. The observed performance for each tag set is reported in Table 5.6.

The 2,000 audio files, for testing a whole audio file, are tagged 5,246 times with the top

Table 5.7: SE-CNN - Top 10 Tags Performance

Tag	Presence	Performance	
choir	708	84.3%	77.8%
male vocals	2,007	83.3%	49.3%
guitar	4,279	74.4%	87.3%
cello	457	72.0%	73.3%
flute	951	71.1%	54.6%
violin	1,579	68.5%	73.7%
female vocals	1,758	67.2%	66.7%
harpsichord	791	66.2%	76.0%
beat	2,057	63.6%	40.8%
vocals	2,570	58.9%	42.3%

■ Recall ■ Precision

50 tags in total. The model trained for the top 50 tags predicts 1.7 tags on average and achieve a ROC-AUC of 87.5%, a micro-averaged recall of 39.6% and a micro-averaged precision with 61.5%. The slightly better performance than in the sample-level test result is a logical consequence, since we have the same ground truth at the audio file level as at the sample-level, but in case of a whole audio file we use more mel-spectrograms for prediction. The result for the ten tags performing the best are reported in Table 5.7. Almost all tags that perform poorly, with recall below 10%, are those tags that occur rarely (less than 1%) in the training data, such as *acoustic*, *funky*, *no guitar*, *no drums*, *bells*, *electric guitar* or *percussion*. The exception is *strange*, *soft*, *quiet*, *synthesizer* and *solo* which are in the middle range in terms of frequency in the training data and *no vocals* which is the 6th most frequently tagged characteristic. In general, tags related to an instrument obtain good performance in relation to their frequency in the training data, while the tempo characteristics *fast* and *slow* perform worse. For the models trained on the top 80 and all 108 tags the top 50 tags achieve similar performance than in the top 50 set, while all the additional characteristics perform poorly with less than 10% recall. This was to be expected, as we observe a significantly decrease of the macro-averaged performances the more tags are used.

Table 5.8: Short-Chunk CNN - Performance

Top x Tags	ROC-AUC	Accuracy	Averaging	Precision	Recall
50	88.5%	95.7%	micro	62.4%	38.9%
			macro	37.8%	25.0%
80	88.4%	97.1%	micro	65.4%	34.5%
			macro	36.0%	18.4%
108	88.0%	94.5%	micro	66.0%	29.3%
			macro	23.1%	12.3%

Short-Chunk CNN

The Short-Chunk CNN achieves the best performance for a batch size of 16. The best observed results for each tag set are shown in Table 5.8.

For the audio file level test on the separated 2,000 audio files, the model trained for the top 50 tags predicts 3.1 tags on average per song. It achieves a ROC-AUC score of 90.0%, 49.0% micro-averaged precision and 58.7% micro-averaged recall. The result for the ten best-performing tags are presented in Table 5.9. As the SE-CNN, also the Short-Chunk CNN achieves better performance for instruments than for characteristics like, *slow*, *fast*, *loud*, *quiet* and *soft*. Again, most characteristics that perform very poorly are those that are tagged rarely. Exceptions are *strange*, *modern* and *trance* which are in the middle field in terms of frequency, as well as *no vocals* and *singer* from the upper field. Also for the Short-Chunk CNN, the model trained for 80 and 108 tags predicted similar tags with good performance in a similar order and can not provide new tags that achieve a reasonable performance.

FCN-4

The FCN-4 model performs best for a batch size of 64. The observed performance for each tag set is reported in Table 5.10.

FCN-4 achieves a good ROC-AUC value of 87.8% for the 2,000 separated audio files. Nevertheless, it only predicts 0.6 tags per song on average and achieves a micro-averaged precision score of 64.6% and micro-averaged recall score of 14.3%. It achieves a recall greater than 0% for 19 tags, and achieves only for three tags a recall above 50%. Precision and recall scores for the ten best-performing tags with FCN-4 are given in Table 5.11. It is worth mentioning that FCN-4 achieves good performance for the characteristics *loud* and *fast*, which are difficult to predict for other models.

VGG-CNN

The VGG-CNN model achieves the best performances for a batch size of 128. The results for each tag set are presented in Table 5.12.

Table 5.9: Short-Chunk CNN - Top 10 Tags Performance

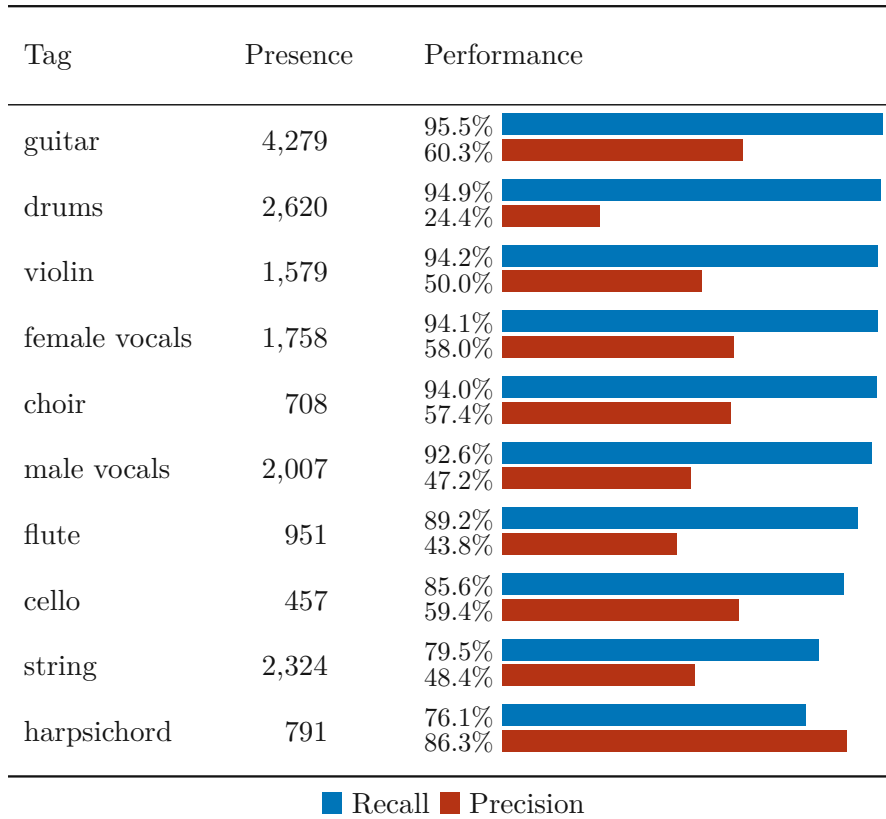


Table 5.10: FCN-4 - Performance

Top x Tags	ROC-AUC	Accuracy	Averaging	Precision	Recall
50	84.2%	95.0%	micro	58.9%	13.8%
			macro	28.6%	7.7%
80	82.5%	96.7%	micro	58.5%	13.1%
			macro	15.9%	4.1%
108	82.3%	97.5%	micro	58.3%	13.1%
			macro	14.7%	3.9%

5. EXPLAINING GENRE CLASSIFICATION WITH SEMANTIC DESCRIPTORS

Table 5.11: FCN-4 - Top 10 Tags Performance

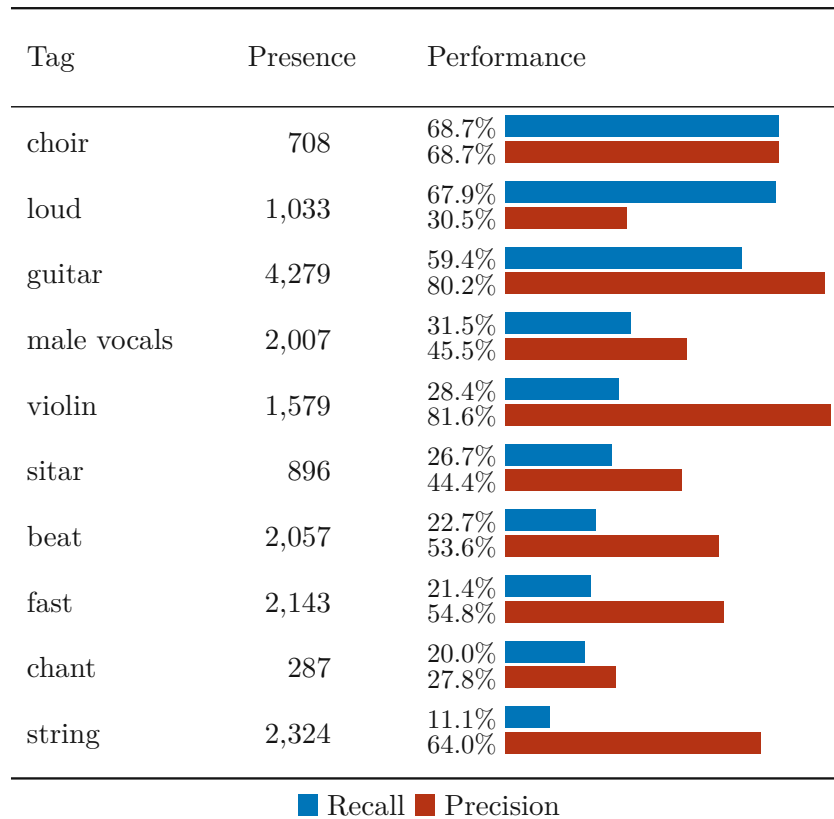












Table 5.12: VGG-CNN - Performance

Top x Tags	ROC-AUC	Accuracy	Averaging	Precision	Recall
50	88.3%	95.6%	micro	64.1%	36.9%
			macro	44.4%	24.0%
80	88.4%	97.1%	micro	66.0%	31.3%
			macro	31.9%	15.6%
108	87.2%	97.7%	micro	64.4%	31.9%
			macro	27.9%	12.0%

Table 5.13: VGG-CNN - Top 10 Tags Performance

Tag	Presence	Performance	
choir	708	92.8% 58.3%	
violin	1,579	90.2% 64.8%	
female vocals	1,758	90.0% 67.7%	
guitar	4,279	88.6% 66.6%	
male vocals	2,007	83.3% 44.7%	
cello	457	81.4% 60.4%	
harpsichord	791	75.5% 82.5%	
vocals	2,570	71.6% 41.6%	
flute	951	71.1% 67.1%	
sitar	896	66.7% 51.2%	

■ Recall ■ Precision

For the audio file level test set, the VGG-CNN model trained on the top 50 tags predicts 2.6 tags on average for each audio file. It achieves a ROC-AUC score of 90.4%, micro-averaged precision of 55.3% and micro-averaged recall of 54.0%. Table 5.13 shows precision and recall for the ten best-performing tags. Also for this model the most tags that perform poorly, with recall below 10%, are tags which are rarely present in the training data, such as *acoustic*, *oriental*, *dark*, *funky*, *no guitar*, *no drums*, *bells* or *electric guitar*. The exception is *trance*, *modern*, *strange* and *no piano*, which are in the middle range in terms of occurrence in the training data and *no vocals*, *soft* and *loud* which are in the top 20 of the most frequently tagged characteristics. We obtain no major differences in tag performance and tag order between models trained on the top 80 and all 108 tags. All additional tags in the top 80 and all 108 tag sets can only achieve a recall below 10%.

Musicnn

The Musicnn performs best with a batch size of 64. Table 5.14 shows the performance of Musicnn for each tag set.

Table 5.14: Musicnn - Performance

Top x Tags	ROC-AUC	Accuracy	Averaging	Precision	Recall
50	86.9%	95.6%	micro	68.9%	26.7%
			macro	36.3%	16.5%
80	87.3%	97.0%	micro	70.1%	24.7%
			macro	33.7%	10.6%
108	86.1%	97.7%	micro	66.7%	26.7%
			macro	24.0%	8.4%

The Musicnn model predicts 2.3 tags on average for each song of the audio file level test set for the top 50 tags. It achieves a ROC-AUC score of 87.9%, micro-averaged precision of 49.4% and micro-averaged recall of 43.5%. The results for the ten best-performing tags are shown in Table 5.15. As also observed for the other models, Musicnn performs bad for tags that are tagged rarely in the training dataset. With an exception for the tags *no vocals*, *soft* and *loud*, which have a bad performance despite their frequent occurrence in the train set as it is the case for VGG-CNN. The Musicnn model trained for 80 and 108 tags achieve similar performance for the tags from the top 50 tag set, but do not obtain any performance above 10% recall for the additional tags.

Stacked CNN

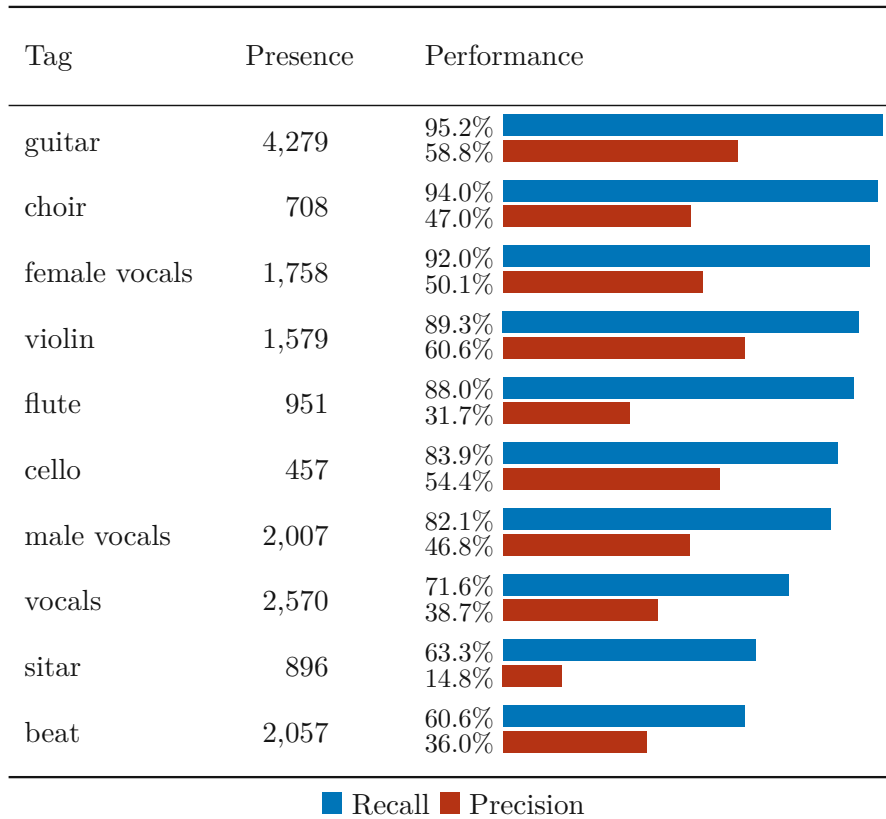
The VGG-CNN model achieves the best performances for batch sizes of 8, 16 and 32. The performance for each tag set is reported in Table 5.16.

Table 5.16: Stacked CNN - Performance

Top x Tags	ROC-AUC	Accuracy	Averaging	Precision	Recall
50	86.0%	94.8%	micro	50.7%	35.7%
			macro	35.99%	30.54%
80	84.7%	96.3%	micro	45.9%	38.1%
			macro	25.0%	21.2%
108	84.3%	96.9%	micro	40.0%	32.3%
			macro	17.5%	15.7%

For the audio file level test set, the Stacked CNN model trained on the top 50 tags predicted 2.2 tags on average. It achieves a ROC-AUC score of 83.4%, micro-averaged precision of 47.7% and micro-averaged recall of 39.8%. The results for the ten best-performing tags can be found in Table 5.17. The most tags that perform poorly, with recall below 10%, are tags which are rarely present in the training data. The exceptions for this model is *trance*, *modern*, *bass*, *strange*, *soft* and *no piano*, which are in the middle range and *no vocals*, *singer* from the top range. The tags *slow*, *fast* and *soft* achieve slightly above 10% in recall and the tags *quiet* and *loud* have with 57.8% and 50.9%

Table 5.15: Musicnn - Top 10 Tags Performance













recall a remarkably good performance compared to other models. Again, the models trained for top 80 and top 108 do not bring up any other characteristics with noteworthy performances.

Summary

We show that overall frequently-tagged characteristics are better to learn, however it should be noted that there are differences in learning ability among the most occurring tags as well. Instruments like *flute*, *cello* or *harpsichord* achieve as good performances as *male vocals*, *female vocals* and *guitar*, although they occur much less frequent. In general, instruments achieve higher performance as compared to tempo and volume tags, like *slow*, *fast*, *loud* and *quiet*, with respect to the frequency of occurrence. Since characteristics are tagged in the course of music games, we have to assume that properties were not clearly defined and brought to a common understanding beforehand. Hence, we assume the characteristics are not annotated with the same quality and therefore it is difficult to assess whether we observe differences in learning ability or inconsistent tagging of audio files.

Table 5.17: Stacked CNN - Top 10 Tags Performance

Tag	Presence	Performance	
violin	1,579	82.9% 50.9%	
cello	457	81.4% 36.0%	
choir	708	80.7% 50.8%	
piano	1,773	79.2% 37.7%	
ambient	1,925	77.4% 17.7%	
drums	2,620	70.5% 30.9%	
female vocals	1,758	69.9% 69.9%	
guitar	4,279	65.6% 76.7%	
quiet	956	57.8% 29.1%	
harpsichord	791	52.0% 92.0%	

■ Recall ■ Precision

An overview of the performance and the average predicted number of tags achieved against the test set on audio file level for all models is presented in Table 5.18. We observed a slightly lower ROC-AUC score of Short-Chunk CNN as compared to VGG-CNN, however recall was higher and the number of predicted tags with 3.1 tags on average was also the highest number of characteristics per song. Only in precision it has worse results to record, but as already described, we attach less importance to this performance metric as it involves the false positive rate. So in terms of learning semantic descriptors we consider the Short-Chunk CNN as the best model.

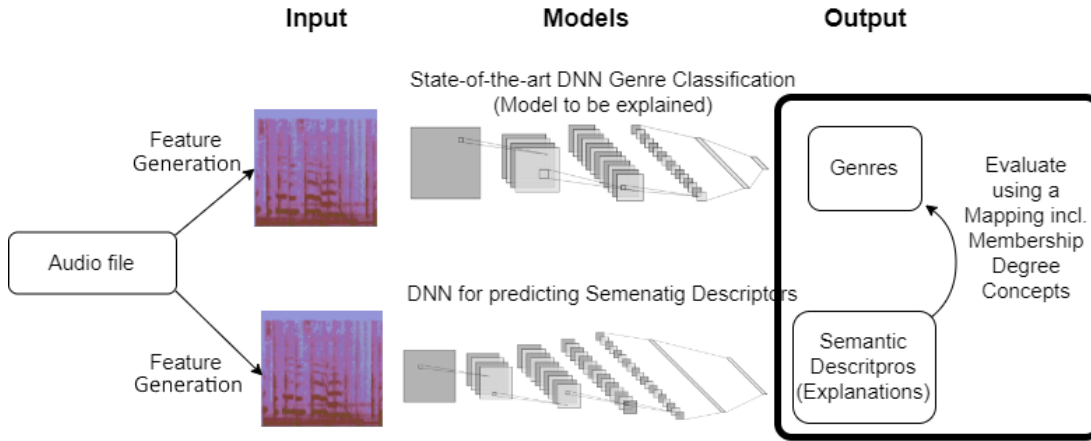
5.2 Mapping between Descriptors and Genre

In this section, we explore how accurate the predicted descriptors from each descriptor learning model, fits the ground truth genre. That is the evaluation of semantic descriptors against genres, shown on the right side of the overview Figure 5.2. Beside the performance of the descriptor model this evaluation is the second assessment of how well the predicted descriptors can be used as an explanation for a genre decision. The performance of these models show how accurate the descriptors fit the audio instances and the mapping

Table 5.18: Model Performances at Audio File Level

Model	avg. Tags	ROC-AUC	Precision	Recall
SE-CNN	1.7	87.5%	61.5%	39.6%
Short-Chunk CNN	3.1	90.0%	49.0%	58.7%
FCN 4	0.6	87.8%	64.6%	14.3%
VGG-CNN	2.6	90.4%	55.3%	54.0%
Musicnn	2.3	87.9%	49.4%	43.5%
Stacked CNN	2.2	83.4%	47.7%	39.8%

Figure 5.2: Schematic description of the relationship between individual components of the explanation system approach



between descriptors and genre show how accurate the descriptors fit the genre. This method of assessing an explanation for a classification decision is based on the publication "Generating Visual Explanations" by Hendricks et al. [HAR⁺16], where they compare the predicted explanations for an image classification of birds species with the definitions of experts. Since there is no such clear definition of music genres, we use the compilation of characteristics for the respective genres. Therefore we create a mapping between music characteristics and genres.

5.2.1 Membership Degree Concept

It is important that the model we use for a mapping from descriptors to genres provides a membership degree for each genre, since we are not interested in a complete decision

Table 5.19: Descriptor - Genre Mapping Results

Model	Ground Truth Genre Affiliation	Recall (Descriptor Learning)
SE-CNN	62.8%	39.6%
Short-Chunk CNN	78.0%	58.7%
FCN 4	32.8%	14.3%
VGG-CNN	73.5%	54.0%
Musicnn	60.8%	43.5%
Stacked CNN	69.5%	39.8%

of which genre a particular set of descriptors is assigned to, but rather in a percentage breakdown of how well a set of descriptors fits each genre.

5.2.2 Implementation

To map descriptors to genres we use a Neural Network, because they are good in pattern recognition, and inferring genre from music characteristics based on the dataset is a pattern recognition task. Furthermore, we can easily assess the membership degree concept by using the softmax activation function in the output layer.

We implement a Neural Network with one hidden layer, a dropout layer and an output layer with softmax activation. For training, we use all audio instances of the MagnaTagATune dataset which are tagged with only one of the genres we are using in our genre classification model and are tagged with at least three music characteristics. We use a train-test split of 75/25. To choose the hyperparameter, we applied grid search for batch size (10, 20, 30), number of units of the hidden layer (50, 100, 200, 400) and the dropout rate (0.3, 0.4, 0.5). The best performance was achieved with batch size of 20, 200 units for the hidden layer and a dropout rate of 0.4 with an accuracy of 80.9%.

We predict the descriptors for the 2,000 separated audio files with each descriptor learning models and obtain the genre membership degree. In a next step, we calculate the average membership degree to the ground truth genre.

5.2.3 Results

The average membership degree to the ground truth genre for each model, together with the performance of the respective model is shown in Table 5.19.

As Short-Chunk CNN achieve the best performance for learning semantic descriptors, and it also obtains the best result for the mapping model between predicted descriptors and

ground truth genres, we consider the Short-Chunk CNN as best state-of-the-art model to learn semantic descriptors of songs and use it to show the visualization of predicted descriptors as well as for our explanation system workflow demonstration.

5.3 Visualization of Descriptors

In this section we describe how the explanation system visualizes the predicted descriptors in the input spectrogram.

To visualize the predicted descriptors we use Deep Taylor Decomposition (DTD), a pixel-wise decomposition of the decision of a Deep Neural Network [BBM⁺15]. The resulting decomposition shows in a heatmap how relevant parts of the input space are to the overall decision or to a particular output neuron. DTD finds application in explaining Deep Neural Networks, especially in image classification [MLB⁺17] but also has been applied successfully in the visualization of audio features [TGT18].

To apply DTD for the visualization of the predicted descriptors, we use iNNvestigate [ALS⁺19], a Python module for analysing Neural Networks.

As DTD cannot handle sigmoid activation function, we have to adjust the model to output the values of the last layer before the sigmoid activation layer. An example for the visualization of a descriptor is shown in Figure 5.3. The used input spectrogram is shown in Figure 5.3a and the heatmap created by iNNvestigate is shown in Figure 5.3b. To relate the DTD heatmap to the spectrogram, we underlay the heatmap with a grayed version of the spectrogram as shown in Figure 5.3c. Since we create more than one mel-spectrogram for a given audio instance to explain a genre decision, we visualize the found descriptor on each mel-spectrogram section and merge them afterwards. This process is explained in more detail in the following section.

5.4 Workflow and Evaluation

In this section we present the workflow of the final explanation system approach.

The first step of the explanation system is to choose an audio file for which the genre should be predicted and the decision should be explained. The audio is loaded and prepared for the genre classification model, which means in our case, resample to 16kHz, create a mel-spectrogram with a window size of 1024, a hop length of 512 and 128 mel bands and cut this spectrogram in sections of size 128x128, which corresponds to approximately 4 seconds. Then, the genre model is applied to all sections and the mostly predicted genre is determined as the overall predicted genre.

In the next step, the explanation system prepares the audio instance for the descriptor model, that is again, resampling the audio file to 16kHz and creating a mel-spectrogram with a window size of 512, a hop length of 256 and 128 mel bands. Then the mel-spectrogram is cut into 128x230 sections. Next, the descriptors are predicted for each

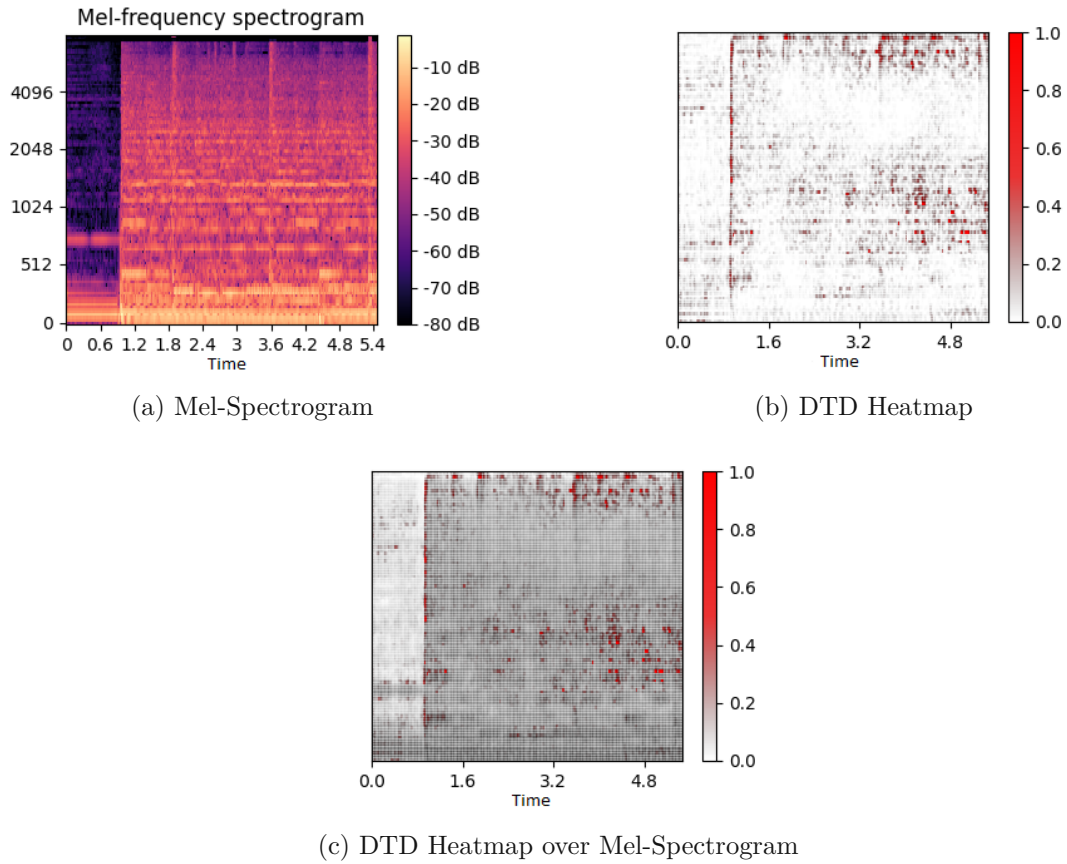


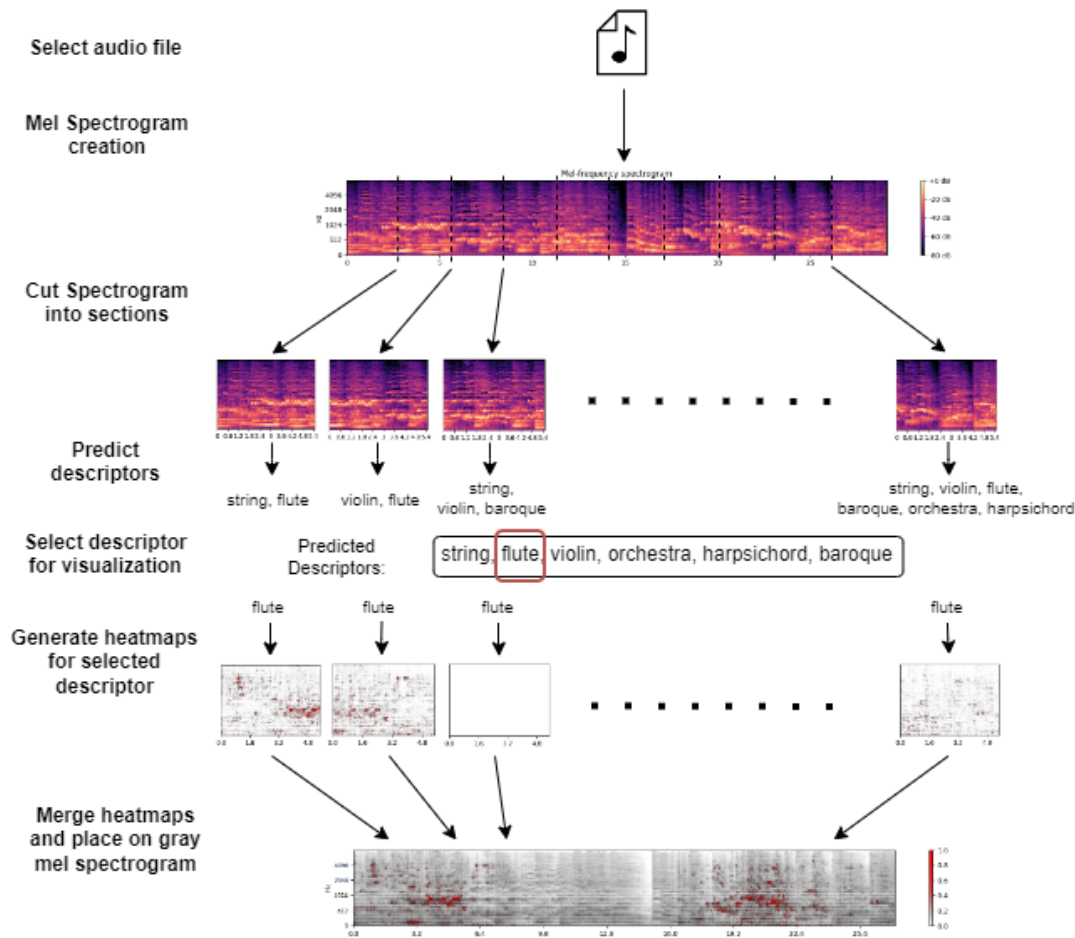
Figure 5.3: Visualization of the tag 'guitar' for the song Samuel P. Huntington from The Seldon Plan

section and stored separately, but also presented all together as the explanation for the genre decision.

Each predicted descriptor can be selected for the visualization. For the visualization the system goes through all mel-spectrograms and calculates the relevance heatmap for all those mel-spectrograms where the descriptor is predicted, using the iNNvestigate module. In a next step, all the heatmaps are merged, whereby a matrix consisting of zeros is used as heatmap for sections where the descriptor is not predicted. On overlapping areas of two heatmaps, the maximum values are used. The resulting merged heatmap is placed over a gray version of the mel-spectrogram.

An overview of this process is shown in Figure 5.4, using the audio clip Sonata II from the Ensemble Mirable. The genre decision to be explained for this example is *classic*, which also corresponds to the ground truth. The explanation system generates nine 128x230 mel-spectrogram sections for which the descriptors *string*, *flute*, *violin*, *orchestra*, *harpsichord* and *baroque* were predicted. The figure also shows that the visualization is

Figure 5.4: Overview of predicting and visualizing descriptors for a audio file



created based on the mel spectrogram sections but is merged afterwards to provide the visualization with the same dimension as the input spectrogram.

The state-of-the-art in automatic music tagging achieves a ROC-AUC of 91% for the top 50 tags of the MagnaTagATune dataset. Taking into account that 12 of the original top 50 tags are genres, that we had to replace with less frequently occurring characteristics, the obtained ROC-AUC of 90% for the explanation system compares well with the state-of-the-art, which answers research question 1. Viewed as an explanatory system. However, these values have to be considered more critically. 58.7% recall means that only 58.7% of the ground truth descriptors are recognized. Also 3.1 predicted descriptors on average is not enough to justify a genre classification, especially if the most predictable descriptors are instruments as well as male and female vocals, which can be representative for any genre. Characteristics which would be more representative for genres like tempo,

emotion and timbre perform worse, however, it can not be clearly stated if it is because they are more complex to learn or because of the nature of the annotation of the used dataset.

To answer research question 2 we have to consider the precision score and the affiliation rate to the ground truth genre. The precision value of 49% is interpreted as more than 50% of the predicted descriptors being incorrect, however, since the descriptor of the genre model shows a 78% affiliation to the ground truth genre also the predicted descriptions, which are not annotated in the ground truth, have to fit well with the actual genre. Thus, we can assume, that part of the supposedly incorrectly predicted descriptors actually describe the song.

CHAPTER 6

Assigning Semantic Descriptors to Feature Maps

In this chapter, we investigate the extent to which it is possible to assign semantic descriptors to feature maps generated by a CNN. Therefore, we compute the intermediate outcome for each convolutional layer of the model to be explained and divide them into the individual feature maps. Then, we train a model with these feature maps for a set of songs as input and music characteristics of the songs as output. The aim of this process is to determine if there are feature maps that perform better than others for a specific music characteristic. This would mean that they extract the semantic information of this music characteristic better as compared to other feature maps. As models we consider Random Forest (RF), k-Nearest-Neighbour (k-NN) and Support Vector Machine (SVM).

6.1 Feature Maps

The information contained in a feature map is more and more finely graded after each convolutional layer, as subsequent convolutional layers generate feature maps based on combinations of the previous layers' feature maps. In addition, the max-pooling layers between the convolutional layers lead to feature maps with smaller dimensions after each layer. The combination of convolutional layer with a max-pooling layer is called convolutional block. The model to be explained includes three convolutional blocks. The first generates 64 feature maps of size 64x64, the second 128 feature maps of size 32x32 and the last one 256 feature maps of size 16x16. To illustrate how feature maps change over the layers, we show randomly selected feature maps as examples for each of the three layers for two different songs to also illustrate how different semantics affect the feature maps. The mel-spectrograms for the two different songs are shown in Figure 6.1, the outcomes of the first and the second convolutional block are shown in Figure 6.2 and the outcomes of the last block are shown in Figure 6.3.

6. ASSIGNING SEMANTIC DESCRIPTORS TO FEATURE MAPS

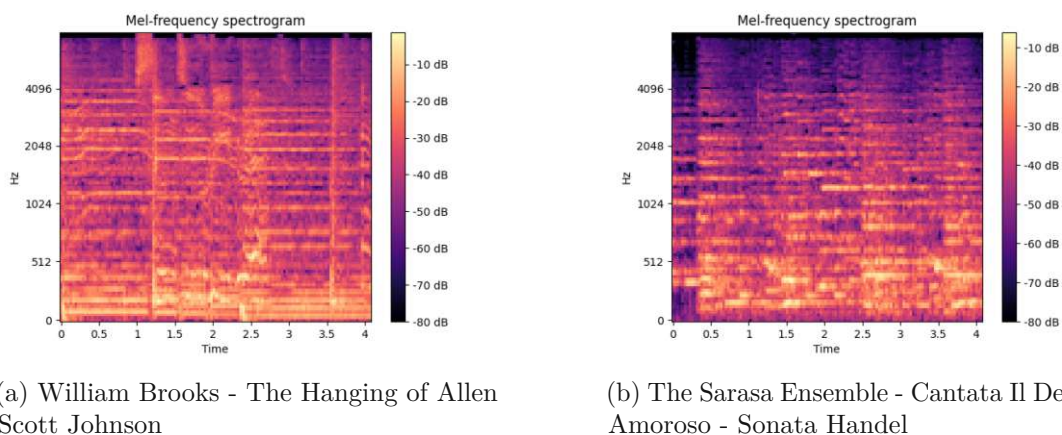


Figure 6.1: Mel-Spectrograms of two different Songs

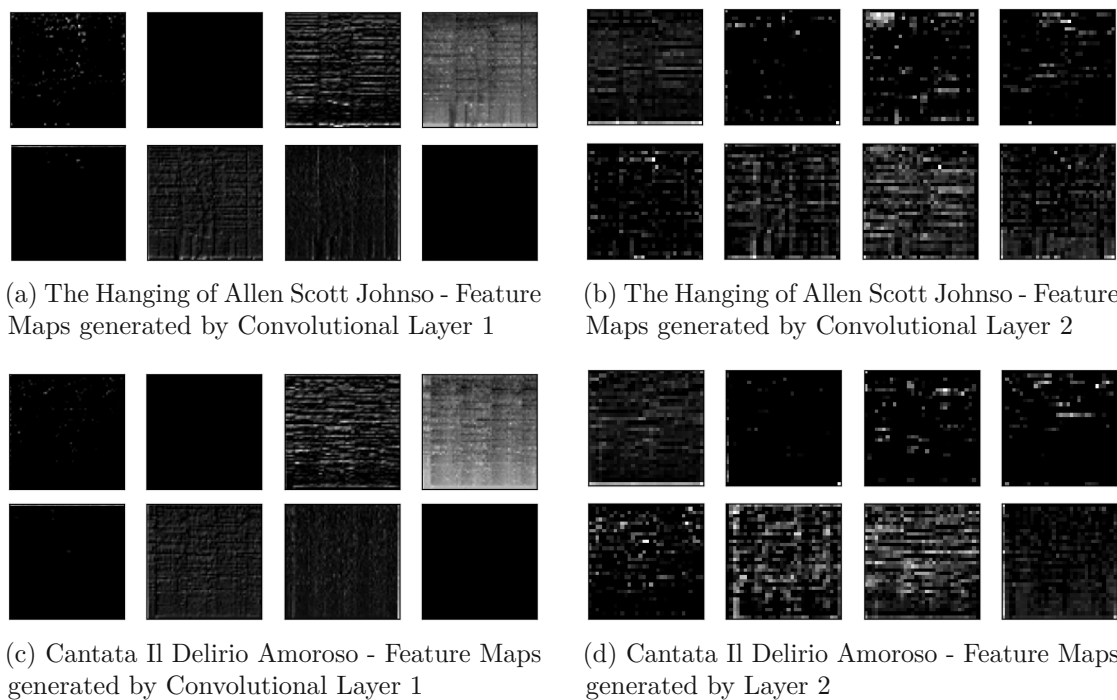
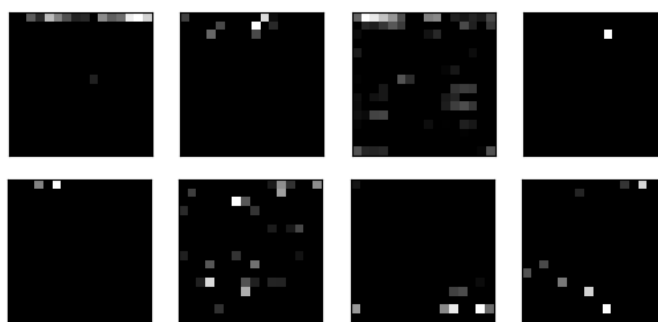
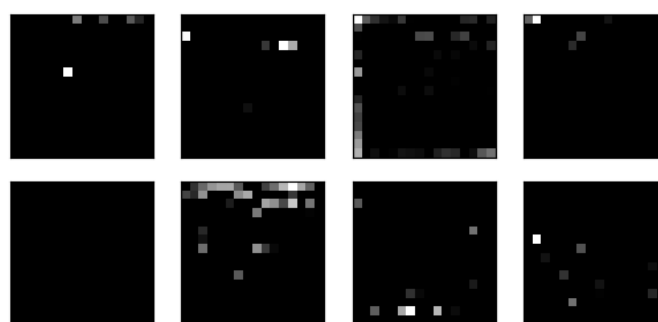


Figure 6.2: Example Feature Maps for two different Songs, generated by the first and second Convolutional Layer



(a) The Hanging of Allen Scott Johnso - Feature Maps generated by Convolutional Block 3



(b) Cantata II Delirio Amoroso - Feature Maps generated by Convolutional Layer 3

Figure 6.3: Example Feature Maps for two different Songs, generated by the third Convolutional Layer

6.2 Data

For this approach, we also use the MagnaTagATune dataset and make use of the pre-processed data from 5.1.1 where synonym tags are already merged and genres are removed. We select ten tags from the 15 most annotated tags namely *guitar*, *string*, *drums*, *fast*, *violin*, *piano*, *synthesizer*, *male vocals*, *slow* and *female vocals*. We discard the tags *beat* and *ambient* as they appear less expressive to us, and *no vocals*, *vocals* and *singer* as we already include *male vocals* and *female vocals*. We select only those audio files that are tagged with at least one of these ten tags, reducing the number of audio files from 18,373 to 14,492. As the most frequently occurring characteristic *guitar* is tagged 4,279 times and the least frequently occurring characteristic *violin* is tagged only 1,579 times, we slightly balance the data by limiting the max occurrence of tags to 3,000. Since the tags are multi-labeled, it was not possible to balance the tags better as this would reduce the less frequently labeled ones to the same extent. The frequency of the tags after balancing can be seen in Table 6.1. We randomly split the data into 80% training data and 20% test data.

Table 6.1: Tag Occurrence in Data used for Assigning Semantic Descriptors to Feature Maps

guitar	string	drums	fast	violin	piano	synthesizer	male vocals	slow	female vocals
3,000	2,071	2,410	2,024	1,536	1,655	1,633	1,801	2,811	1,683

6.3 Model Selection

For the training of the models, we use the Python library Scikit-learn¹. To select a model that performs best for our data, we apply 3-fold cross validation. For k-Nearest-Neighbor, in addition to the default value of k=5, we also consider k=3 and k=7. For Random Forest, the default value for the number of trees of 100 is considered, as well as 200 and 500 trees. The Python module tqdm estimate a runtime of 52 hours for Support Vector Machines for the intermediate outputs of the last layer, after processing the first five of 256 iterations, where the last layer already contains the smallest amount of data. Considering that we would have had to run the model for one setting for each layer and thus 3 times we decided to exclude SVM.

To be able to select one model that performs best across all characteristics, we average the F1-score across all characteristics. However, the 3-Nearest-Neighbor model achieve the highest average F1-score, but also achieve the best F1-score for almost each individual characteristic. Thus, we choose 3-Nearest-Neighbor as model for this approach and trained it for the full training set and evaluated it on the test set. The detailed result of the cross validation can be found in Appendix A.

6.4 Results

In this section, we present the best and worst performing feature maps for each characteristic and visualize them for two example songs.

6.4.1 First Convolutional Layer

The highest five performances of feature maps of the first convolutional layer (see Table 6.2), shows that the feature map with index 2 is present for almost all characteristics. Other feature maps are also present more than once, such as for example the feature map with index 50. This can be explained by the fact, that at this point only one filter is applied to the input for each feature map, and that there are feature maps that have more information filtered out than others. Especially in combination with the table showing the worst five performing feature maps (Table 6.3). One can see that after the first convolutional layer there are already feature maps focusing on different information. An example for this focus is the feature map with index 20, which perform quite good with an F1-score of 29.8% for the tag *guitar* (see Table 6.3) but performs poor for the

¹<https://scikit-learn.org/>

Table 6.2: Top 5 Feature Map Performance - Convolutional Layer 1

guitar			string			drums			fast		
FM	Acc	F1	FM	Acc	F1	FM	Acc	F1	FM	Acc	F1
2	78.3%	33.4%	2	83.1%	35.9%	51	73.8%	46.0%	51	79.0%	45.3%
18	51.2%	31.2%	41	78.3%	30.4%	37	72.8%	45.6%	23	76.9%	43.6%
20	64.3%	29.8%	50	81.0%	26.2%	10	69.0%	44.3%	10	80.6%	43.4%
30	55.8%	29.6%	4	80.7%	25.8%	16	70.2%	43.1%	54	80.2%	43.3%
27	48.1%	28.3%	59	81.1%	25.6%	24	76.0%	43.0%	16	82.3%	42.0%

violin			piano			synthesizer			male vocals		
FM	Acc	F1	FM	Acc	F1	FM	Acc	F1	FM	Acc	F1
41	81.7%	40.5%	2	89.0%	33.5%	11	84.4%	23.5%	2	86.9%	25.2%
2	87.8%	37.6%	5	65.7%	30.1%	1	83.0%	23.5%	32	82.0%	24.1%
15	85.3%	27.1%	62	85.2%	30.0%	30	86.0%	23.4%	31	86.5%	20.4%
50	87.0%	27.0%	35	73.2%	29.6%	52	77.8%	23.3%	50	86.3%	18.6%
48	85.6%	26.8%	31	88.2%	28.6%	39	84.6%	23.2%	10	86.4%	15.0%

slow			female vocals		
FM	Acc	F1	FM	Acc	F1
25	57.6%	39.8%	2	87.4%	10.3%
24	68.1%	39.5%	31	87.6%	9.9%
16	67.4%	39.4%	51	87.6%	9.4%
48	64.3%	39.2%	56	86.6%	9.2%
2	73.1%	38.7%	1	87.6%	8.8%

tag *violin* with an F1-score of 0.6% (see Table 6.3). Similar behaviour can be observed for other feature maps and properties. The characteristic *slow* achieved comparably good performance for all feature maps, based on what we can conclude that this feature is good to learn for a light abstraction level.

6.4.2 Second Convolutional Layer

Even though the performance of the best-performing feature maps after the second convolutional layer, deteriorate for almost all features as compared to the first layer, we can also see that the performances of the five worst performing feature maps are going towards zero, which shows that the feature maps are focusing more and more on different information. This specialization is also reflected in the fact, that after the second convolutional layer, there is no longer a feature map that performs well for many characteristics. The top performing feature maps for each tag are shown in Table 6.4 and the worst performing feature maps for each tag are shown in Table 6.5.

6.4.3 Third Convolutional Layer

Also for the feature maps of the third convolutional layer, we observed a deterioration regarding the performances of the top five feature maps of each tag. However, this implies

6. ASSIGNING SEMANTIC DESCRIPTORS TO FEATURE MAPS

Table 6.3: Bottom 5 Feature Map Performance - Convolutional Layer 1

guitar			string			drums			fast		
FM	Acc	F1	FM	Acc	F1	FM	Acc	F1	FM	Acc	F1
61	73.6%	16.1%	5	83.7%	6.0%	40	79.5%	16.9%	43	82.6%	16.2%
44	72.4%	15.9%	16	83.4%	5.0%	45	79.5%	16.7%	6	79.7%	15.7%
4	76.1%	15.1%	8	82.4%	3.5%	43	79.9%	16.0%	8	82.6%	15.6%
41	77.2%	14.7%	40	82.2%	2.6%	27	81.8%	15.6%	45	82.7%	15.0%
28	73.4%	10.3%	27	82.5%	0.9%	28	78.9%	14.9%	27	82.8%	8.5%

violin			piano			synthesizer			male vocals		
FM	Acc	F1	FM	Acc	F1	FM	Acc	F1	FM	Acc	F1
11	87.7%	1.9%	27	86.7%	7.8%	62	78.6%	13.3%	3	85.0%	3.6%
21	87.6%	1.9%	20	88.2%	7.6%	5	84.9%	12.5%	18	86.0%	2.2%
40	87.1%	1.8%	18	87.6%	6.6%	35	84.0%	9.6%	28	85.9%	2.2%
20	87.5%	0.6%	11	88.3%	4.6%	27	86.7%	6.8%	6	85.3%	1.6%
27	87.2%	0.0%	30	87.6%	3.7%	32	86.8%	6.3%	27	85.8%	0.6%

slow			female vocals		
FM	Acc	F1	FM	Acc	F1
11	76.9%	30.8%	62	87.7%	0.6%
19	66.8%	30.1%	27	87.5%	0.0%
33	64.6%	29.2%	40	87.6%	0.0%
59	65.4%	29.2%	43	87.5%	0.0%
27	69.9%	21.4%	61	87.6%	0.0%

Table 6.4: Top 5 Feature Map Performance - Convolutional Layer 2

guitar			string			drums			fast		
FM	Acc	F1	FM	Acc	F1	FM	Acc	F1	FM	Acc	F1
46	51.6%	34.6%	10	79.5%	29.2%	76	76.9%	38.9%	90	80.3%	40.9%
73	63.7%	31.5%	3	75.2%	28.9%	90	78.1%	38.7%	21	79.9%	40.8%
71	76.9%	31.4%	80	81.4%	26.9%	23	78.8%	38.0%	73	78.6%	38.8%
99	76.9%	31.2%	97	82.3%	25.1%	67	79.0%	37.9%	39	79.7%	38.3%
39	43.3%	31.2%	81	69.2%	24.8%	112	68.7%	37.9%	112	76.4%	38.0%

violin			piano			synthesizer			male vocals		
FM	Acc	F1	FM	Acc	F1	FM	Acc	F1	FM	Acc	F1
10	86.2%	36.2%	65	83.4%	27.1%	67	83.7%	23.7%	37	83.1%	26.5%
7	85.0%	28.1%	83	75.6%	24.9%	71	81.3%	22.6%	65	81.0%	25.2%
97	86.6%	24.8%	93	83.2%	23.4%	36	83.4%	22.5%	84	81.5%	24.8%
80	85.2%	24.2%	63	83.4%	23.4%	90	82.6%	22.5%	78	78.5%	23.2%
108	84.5%	23.1%	32	81.7%	23.2%	10	83.2%	21.7%	38	81.0%	22.8%

slow			female vocals		
FM	Acc	F1	FM	Acc	F1
77	63.3%	39.3%	99	86.7%	13.1%
30	63.6%	38.9%	7	85.6%	13.0%
25	67.3%	38.1%	53	86.9%	12.8%
96	61.7%	38.0%	71	86.4%	12.0%
7	66.6%	37.0%	64	86.6%	11.7%

Table 6.5: Bottom 5 Feature Map Performance - Convolutional Layer 2

guitar			string			drums			fast		
FM	Acc	F1	FM	Acc	F1	FM	Acc	F1	FM	Acc	F1
122	77.4%	9.6%	86	83.3%	1.9%	3	80.6%	2.8%	62	81.7%	1.3%
121	75.3%	8.9%	11	83.2%	1.4%	57	79.6%	2.7%	3	81.8%	0.9%
43	73.6%	8.6%	45	82.9%	1.4%	74	80.8%	2.4%	31	81.9%	0.9%
89	76.9%	6.5%	95	82.8%	1.4%	72	81.1%	1.7%	56	82.0%	0.9%
16	77.9%	5.8%	8	83.8%	0.5%	118	81.2%	0.8%	118	82.2%	0.4%

violin			piano			synthesizer			male vocals		
FM	Acc	F1	FM	Acc	F1	FM	Acc	F1	FM	Acc	F1
8	87.6%	0.0%	39	88.5%	2.0%	86	85.0%	3.6%	46	86.0%	0.6%
11	87.6%	0.0%	11	88.2%	1.3%	74	86.6%	2.9%	74	85.9%	0.6%
12	87.5%	0.0%	86	87.6%	1.3%	46	87.2%	1.2%	22	85.5%	0.6%
46	87.6%	0.0%	12	88.2%	0.0%	118	87.0%	1.2%	34	85.8%	0.0%
73	87.6%	0.0%	73	88.4%	0.0%	55	86.8%	0.6%	118	85.9%	0.0%

slow			female vocals		
FM	Acc	F1	FM	Acc	F1
24	79.3%	19.7%	11	87.6%	0.0%
100	78.6%	18.1%	46	87.6%	0.0%
39	77.4%	16.3%	55	87.5%	0.0%
86	80.5%	15.0%	83	87.4%	0.0%
12	77.8%	7.4%	114	87.3%	0.0%

that the CNN abstracts more detailed information per feature map across layers, but not in such a way that a single feature map can represent the entirety of any of our selected semantic characteristics. The top performing feature maps produced by the third convolutional layer are shown in Table 6.6 and the worst performing feature maps in Table 6.7.

6.4.4 Visualization

To show how the top performing feature maps for the characteristics looks like, we visualize the same feature maps of each convolutional layer for two different songs and highlight the best performing feature map for the respective tagged characteristics. We take one audio file tagged as *metal* (Magnatune Compilation - Pizzle in my livid eyes), for which we highlight the characteristics *guitar*, *drums* and *male vocal*, and one audio file tagged as *classic* (Jami Sieber - Prayer), for which we highlight the characteristics *piano*, *string* and *violin*. The feature maps for the examples, generated by the first convolutional layer are shown in Figure 6.4, those generated by the second layer are shown in Figure 6.5 and those generated by the last layer are shown Figure 6.6. In this example, one can clearly see how the appearance of the top performing feature maps for the respective characteristics increasingly differs across the convolutional layers.

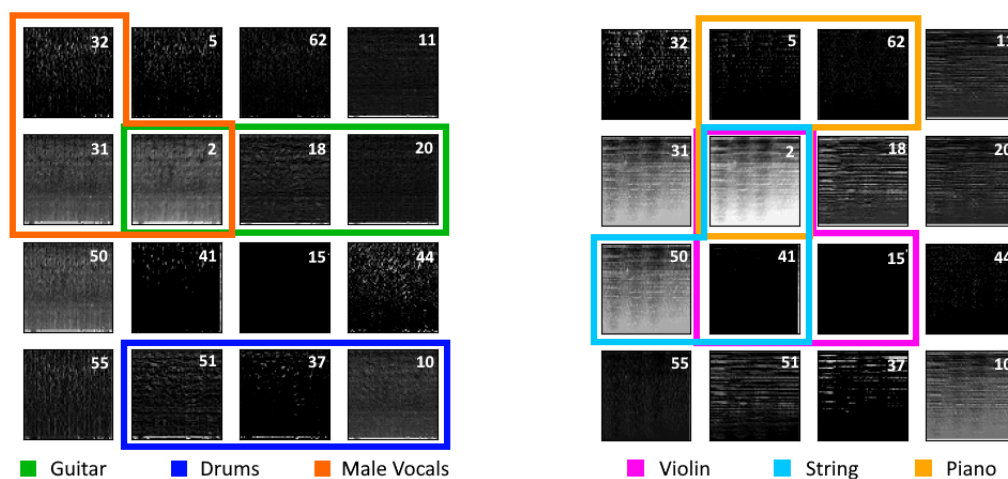
6. ASSIGNING SEMANTIC DESCRIPTORS TO FEATURE MAPS

Table 6.6: Top 5 Feature Map Performance - Convolutional Layer 3

guitar			string			drums			fast		
FM	Acc	F1	FM	Acc	F1	FM	Acc	F1	FM	Acc	F1
68	77.1%	29.6%	198	78.6%	29.4%	57	61.1%	42.7%	216	71.9%	49.2%
184	74.8%	28.5%	96	80.8%	28.6%	78	63.9%	42.2%	191	74.3%	46.6%
124	77.5%	28.4%	111	80.9%	24.9%	215	53.2%	42.1%	11	67.4%	46.3%
135	53.0%	27.9%	156	81.4%	24.6%	24	61.3%	41.0%	78	68.0%	46.0%
234	77.1%	27.4%	131	80.0%	22.9%	18	59.5%	40.9%	110	64.3%	44.6%

violin			piano			synthesizer			male vocals		
FM	Acc	F1	FM	Acc	F1	FM	Acc	F1	FM	Acc	F1
198	83.5%	28.5%	198	85.3%	33.2%	124	82.7%	23.7%	205	84.0%	24.6%
96	84.9%	27.1%	192	86.1%	26.0%	241	75.6%	23.6%	124	85.3%	24.3%
81	85.5%	25.8%	152	85.8%	24.7%	219	83.6%	22.4%	95	85.7%	24.3%
133	85.7%	25.2%	8	85.7%	23.6%	49	82.7%	21.8%	219	84.8%	23.1%
126	85.8%	24.3%	96	84.9%	23.5%	83	82.2%	21.6%	128	81.3%	23.1%

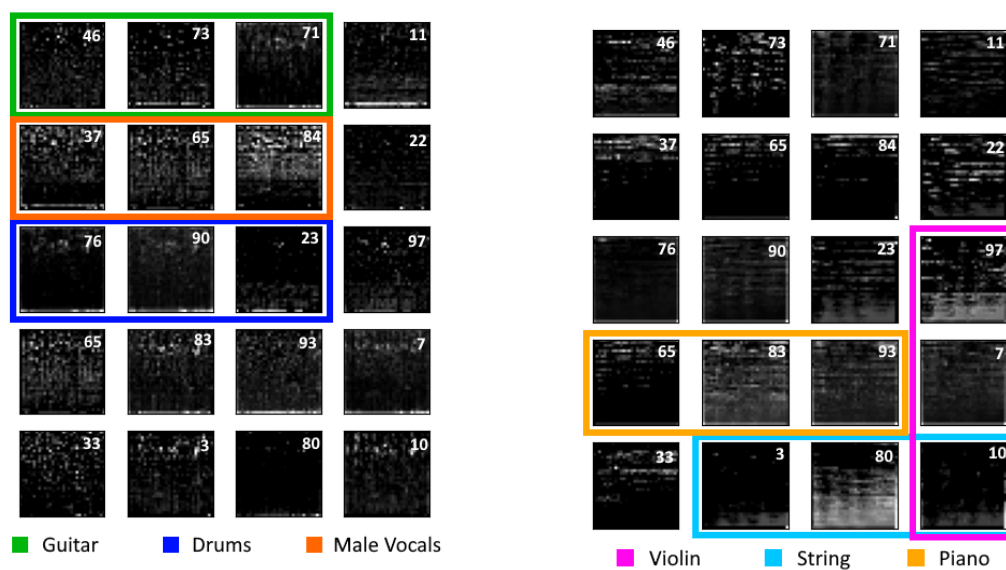
slow			female vocals		
FM	Acc	F1	FM	Acc	F1
131	75.7%	36.1%	185	76.0%	19.2%
101	61.9%	36.0%	1	40.0%	18.1%
156	73.2%	36.0%	182	86.3%	17.0%
112	73.1%	35.9%	253	77.2%	16.7%
35	75.8%	35.6%	206	86.3%	16.6%



(a) MagnaTune compilation - Pizzile in my livid eyes

(b) Jami Sieber - Prayer

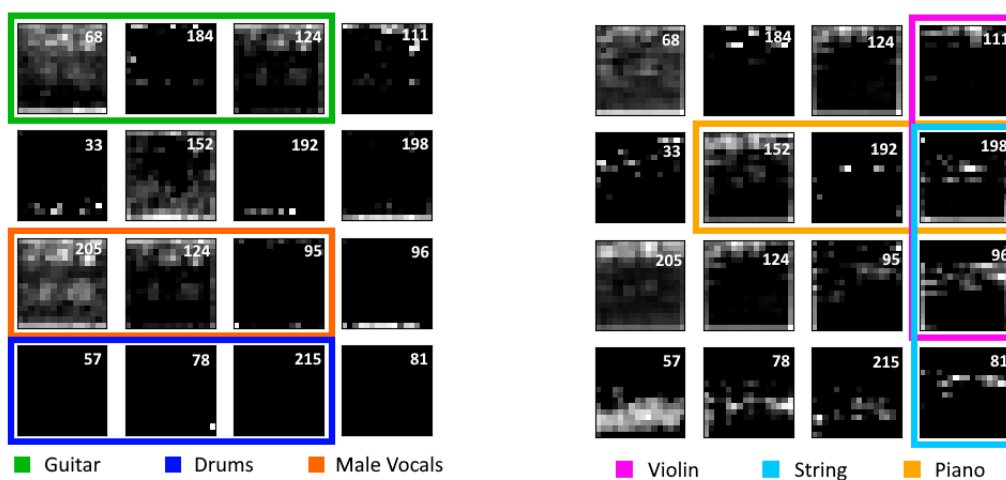
Figure 6.4: Convolutional Layer 1 - Highlighting best performing Feature Maps



(a) Magnatune Compilation - Pizzle in my livid eyes

(b) Jami Sieber - Prayer

Figure 6.5: Convolutional Layer 2 - Highlighting best performing Feature Maps



(a) Magnatune compilation - Pizzle in my livid eyes

(b) Jami Sieber - Prayer

Figure 6.6: Convolutional Layer 3 - Highlighting best performing Feature Maps

Table 6.7: Bottom 5 Feature Map Performance - Convolutional Layer 3

guitar			string			drums			fast		
FM	Acc	F1	FM	Acc	F1	FM	Acc	F1	FM	Acc	F1
1	77.0%	5.6%	132	82.1%	6.3%	23	78.5%	5.0%	12	78.7%	5.0%
10	76.6%	4.3%	86	80.8%	6.2%	31	78.7%	3.3%	1	80.7%	4.8%
53	77.5%	3.8%	34	81.8%	5.8%	92	80.2%	3.1%	239	80.1%	3.9%
245	78.6%	2.9%	247	82.5%	5.2%	221	78.7%	2.9%	97	80.0%	3.5%
20	78.9%	1.9%	137	81.3%	3.3%	119	80.1%	2.0%	18	80.7%	1.6%

violin			piano			synthesizer			male vocals		
FM	Acc	F1	FM	Acc	F1	FM	Acc	F1	FM	Acc	F1
247	86.6%	3.5%	180	86.4%	4.0%	18	86.4%	1.2%	215	85.0%	1.1%
1	86.5%	2.3%	34	86.2%	3.9%	141	86.4%	1.2%	221	84.7%	1.0%
45	86.0%	2.2%	253	85.5%	3.7%	165	85.7%	0.6%	59	84.5%	1.0%
252	87.2%	1.2%	137	87.2%	3.6%	20	87.3%	0.0%	202	85.5%	0.6%
98	86.6%	1.2%	17	86.8%	3.0%	53	86.8%	0.0%	141	85.4%	0.0%

slow			female vocals		
FM	Acc	F1	FM	Acc	F1
154	75.3%	10.7%	215	86.5%	1.8%
144	77.1%	9.2%	144	86.2%	1.7%
122	76.8%	9.1%	108	86.1%	1.7%
89	77.8%	8.0%	20	86.8%	1.2%
245	78.3%	4.6%	53	86.8%	1.2%

6.5 Evaluation

Due to the fact that there are feature maps performing best for one semantic descriptor, but performing worst for other semantic descriptors shows that each feature map learned a different semantic and that the learned semantics of an individual feature map represented some music characteristics better than other. However, as even the best performing feature maps for each characteristic did not achieve an F1-score above 50%, it appears that the model being explained do not learn any specific semantic that is fully comparable to any music characteristic we have selected. Since this is an introspective approach, analyzing a specific model, it does not mean that this can not be the case for other implementations of a CNN based genre classification.

Conclusion and Future Work

In this chapter we conclude our thesis by summarizing our findings, reviewing our initially raised research questions and give an outlook on future work.

7.1 Conclusion

The aim of this master thesis was the explanation of a state-of-the-art CNN genre classification model and proposed two different approaches. The first proposed approach aims to justify a genre decision by providing music characteristics as additional information of the classified song and covers the first two research questions.

Research question 1: *To what extent can semantic descriptors be learned from audio inputs as used for a state-of-the art CNN genre classification model?*

The prediction of the semantic descriptors by our proposed explanation system achieves a ROC-AUC of 90%, and thus competes with state-of-the-art music auto tagging in terms of performance, which answers research question 1. However, considering that the descriptors should serve as explanation for a genre decision, a recall of 58.7%, precision of 49% and 3.1 predicted descriptors on average is not sufficient. A detailed presentation of the overall performance as well as performances on different tags can be found in Section 5.1.7 for all the models investigated, whereby the performance stated here refers to the best-performing model, namely the Short-Chunk CNN.

Research question 2: *To what extent are the explanations in form of semantic descriptors, provided by the explanation system, relevant to the ground truth genre?*

To answer research question 2 we proposed a Neural Network along with membership degree concepts to map semantic descriptors to genre affiliation which is described in Section 5.2. The precision score of 49% is interpreted as 51% of the predicted descriptors does not match the ground truth descriptors, however, our descriptor to genre mapping

model shows 78% affiliation to the ground truth genre for the predicted descriptions, which leads us to the conclusion that part of the supposedly incorrectly predicted descriptors actually describe the song.

In the second approach, semantic descriptors are used to gain introspective understanding of a state-of-the-art genre classification model. The semantic descriptors are assigned to the feature maps of the CNN that covers the semantic information best. For this assignment, k-NN and Random Forest were considered, whereby the former performed better.

Research question 3: *To what extent can semantic descriptors be assigned to feature maps of a state-of-the-art CNN genre classification model*

The results of the different layers as well as the visualization of the feature maps clearly show how the represented semantics from a feature map becomes more detailed across the convolutional layers. We observed differences in how well feature maps can represent particular characteristics between different feature maps, but no feature map was found that could fully capture a characteristic. However, this approach provides a way to improve the understanding of the convolutional layers of a genre classification model. The five best-performing and the five worst-performing feature maps for all semantic descriptors investigated are listed in Tables 6.2-6.7 along with the accuracy and F1 score and are interpreted in more detail in Section 6.4.

7.2 Future Work

We introduced a justification explanation approach for CNN based music genre classification and obtained performances similar to the state-of-the-art in the music auto-tagging domain. However, with 3.1 predicted descriptors on average per genre classification and additionally, mainly instruments predicted with acceptable performance, the justification explanation approach is not yet ready to be used. Future work could investigate, if semantic descriptors, which are sparsely or not at all annotated in the MagnaTagATune dataset, can be learned from the Last.FM dataset. Also splitting the descriptor model into multiple models by selecting the model with the best performance for each tag, rather than the model with the best average performance, can be considered.

Our approach to assign semantic descriptors to feature maps generated by the convolution layers within the genre classification model showed that some feature maps can represent particular semantic descriptors better than others. However, we could not determine a single feature map which fully represents a descriptor. Future work could analyse deeper models to investigate how the feature maps evolve across more layers.

In general, a music database which is annotated less sparsely or even annotated completely with music characteristics, such that a not annotated characteristic actually means this characteristic is not included, could improve our approaches. Since we are not using mel-spectrograms from the whole audio files but cut them into smaller sections, a database with more finely granulated annotations could improve the approaches as well.

List of Figures

2.1	Schematic description of the relationship between individual components of the explanation system approach	8
2.2	Overview of Assigning Descriptive Labels to Feature Maps Approach . . .	9
3.1	Example of a 128x128 Mel-Spectrogram for the song 'Howl at the Moon' from Seismic Anamoly	14
3.2	Validation loss fluctuation for CNN following [CKN ⁺ 19] with and without using Keras' callback methods.	18
3.3	Architecture of the selected CNN Genre Classification Model to be Explained (created with the online-tool NN-SVG [LeN19])	19
5.1	Tag Histogram of the 108 remaining Tags after Pre-Processing the MagnaTagATune Dataset	31
5.2	Schematic description of the relationship between individual components of the explanation system approach	49
5.3	Visualization of the tag 'guitar' for the song Samuel P. Huntington from The Seldon Plan	52
5.4	Overview of predicting and visualizing descriptors for a audio file	53
6.1	Mel-Spectrograms of two different Songs	56
6.2	Example Feature Maps for two different Songs, generated by the first and second Convolutional Layer	56
6.3	Example Feature Maps for two different Songs, generated by the third Convolutional Layer	57
6.4	Convolutional Layer 1 - Highlighting best performing Feature Maps . . .	62
6.5	Convolutional Layer 2 - Highlighting best performing Feature Maps . . .	63
6.6	Convolutional Layer 3 - Highlighting best performing Feature Maps . . .	63

List of Tables

2.1	Comparison of the data sets regarding properties relevant for the master thesis	12
3.1	Number of created Mel-Spectrograms for each Genre for Balanced and Semi-Balanced Data	15
3.2	Layer listing of Genre Classification Model following [CKN ⁺ 19]	16
3.3	Layer listing of Genre Classification Model following [PG20]	17
3.4	Music Genre Classification CNNs - Performance	18
4.1	Systematic Literature Review: Remaining papers after first iteration . . .	24
4.2	Systematic Literature Review: Remaining papers after elimination based on abstract	25
4.3	Systematic Literature Review: Resulting models and their performance for the top 50 tags of MagnaTagATune	28
5.1	Equivalent Tags of MagnaTagATune	30
5.2	Detailed description of SE-CNN Model [JHP22] (for 108 tags)	33
5.3	Detailed description of Short-Chunk CNN Model [WFBS20] (for 108 tags)	34
5.4	Detailed description of FCN-4 Model [CFS16] and the adaption based on [FBJ ⁺ 21] (for 108 tags)	36
5.5	Detailed description of Stacked CNN Model [dSdCD20] (for 108 tags) . .	39
5.6	SE-CNN - Performance	40
5.7	SE-CNN - Top 10 Tags Performance	41
5.8	Short-Chunk CNN - Performance	42
5.9	Short-Chunk CNN - Top 10 Tags Performance	43
5.10	FCN-4 - Performance	43
5.11	FCN-4 - Top 10 Tags Performance	44
5.12	VGG-CNN - Performance	44
5.13	VGG-CNN - Top 10 Tags Performance	45
5.14	Musicnn - Performance	46
5.16	Stacked CNN - Performance	46
5.15	Musicnn - Top 10 Tags Performance	47
5.17	Stacked CNN - Top 10 Tags Performance	48
5.18	Model Performances at Audio File Level	49
5.19	Descriptor - Genre Mapping Results	50
		69

6.1	Tag Occurrence in Data used for Assigning Semantic Descriptors to Feature Maps	58
6.2	Top 5 Feature Map Performance - Convolutional Layer 1	59
6.3	Bottom 5 Feature Map Performance - Convolutional Layer 1	60
6.4	Top 5 Feature Map Performance - Convolutional Layer 2	60
6.5	Bottom 5 Feature Map Performance - Convolutional Layer 2	61
6.6	Top 5 Feature Map Performance - Convolutional Layer 3	62
6.7	Bottom 5 Feature Map Performance - Convolutional Layer 3	64
A.1	CV Results - Convolutional Layer 1	78
A.2	CV Results - Convolutional Layer 2	79
A.3	CV Results - Convolutional Layer 3	80

Bibliography

- [ALS⁺19] Maximilian Alber, Sebastian Lapuschkin, Philipp Seegerer, Miriam Hägele, Kristof T. Schütt, Grégoire Montavon, Wojciech Samek, Klaus-Robert Müller, Sven Dähne, and Pieter-Jan Kindermans. Innvestigate neural networks! *Journal of Machine Learning Research*, 20(93):1–8, 2019.
- [BBM⁺15] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS One*, 10(7):e0130140, 2015.
- [BC17] Or Biran and Courtenay Cotton. Explanation and justification in machine learning: A survey. In *IJCAI-17 workshop on explainable AI (XAI)*, volume 8, pages 8–13, 2017.
- [BMEWL11] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.
- [CCJM21] Zhen Chen, Maoyong Cao, Peng Ji, and Fengying Ma. Research on crop disease classification algorithm based on mixed attention mechanism. In *Journal of Physics: Conference Series*, volume 1961, page 012048, 2021.
- [CFS16] Keunwoo Choi, György Fazekas, and Mark B. Sandler. Automatic Tagging Using Deep Convolutional Neural Networks. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR 2016)*, pages 805–811, 2016.
- [CKN⁺19] Snigdha Chillara, AS Kavitha, Shwetha A Neginhal, Shreya Haldia, and KS Vidyullatha. Music genre classification using machine learning algorithms: a comparison. *IRJET International Research Journal of Engineering and Technology*, 6(05):851–858, 2019.
- [CN19] Jeong Choi and Juhan Nam. Zero-shot learning for audio-based music classification and tagging. In *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR 2019)*, pages 67–74, 2019.

- [DBVB17] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. FMA: A dataset for music analysis. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, 2017.
- [dSdCD20] Juliano Donini da Silva, Yandre Maldonado Gomes da Costa, and Marcos Aurélio Domingues. Improving musical tag annotation with stacking and convolutional neural networks. In *Proceedings of the International Conference on Systems, Signals and Image Processing (IWSSIP 2020)*, pages 393–398. IEEE, 2020.
- [FBJ⁺21] Andres Ferraro, Dmitry Bogdanov, Xavier Serra Jay, Ho Jeon, and Jason Yoon. How low can you go? reducing frequency and time resolution in current cnn architectures for music auto-tagging. In *Proceedings of the 28th European Signal Processing Conference (EUSIPCO 2020)*, pages 131–135. IEEE, 2021.
- [FNT⁺20] Jianyu Fan, Eric Nichols, Daniel Tompkins, Ana Elisa Méndez Méndez, Benjamin Elizalde, and Philippe Pasquier. Multi-label sound event retrieval using a deep learning-based siamese structure with a pairwise presence matrix. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2020)*, pages 3482–3486. IEEE, 2020.
- [GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016.
- [GSS20] Anirudh Ghildiyal, Komal Singh, and Sachin Sharma. Music genre classification using machine learning. In *Proceedings of the 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA 2020)*, pages 1368–1372. IEEE, 2020.
- [HAR⁺16] Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. Generating visual explanations. In *Proceedings of the 14th European Conference on Computer Vision (ECCV 2016)*, volume IV, pages 3–19. Springer International Publishing, 2016.
- [HHDA18] Lisa Anne Hendricks, Ronghang Hu, Trevor Darrell, and Zeynep Akata. Grounding visual explanations. In *Proceedings of the 15th European Conference on Computer Vision (ECCV 2018)*, volume II, pages 264–279. Springer International Publishing, 2018.
- [HSS18] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2018)*, pages 7132–7141, 2018.

- [Iza22] Habib Izadkhah. Chapter 7 - introduction to deep learning. In Habib Izadkhah, editor, *Deep Learning in Bioinformatics*, pages 131–174. Academic Press, 2022.
- [JHP22] Chen Ju, Lixin Han, and Guozheng Peng. Music auto-tagging based on attention mechanism and multi-label classification. In *Proceedings of the International Conference on Image, Vision and Intelligent Systems (ICIVIS 2021)*, pages 245–255. Springer Singapore, 2022.
- [KGM⁺21] Dillon Knox, Timothy Greer, Benjamin Ma, Emily Kuo, Krishna Somandepalli, and Shrikanth Narayanan. Loss function approaches for multi-label music tagging. In *Proceedings of the 18th International Conference on Content-Based Multimedia Indexing (CBMI 2021)*, pages 1–4. IEEE, 2021.
- [KSH17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.
- [LA22] Dhevan S. Lau and Ritesh Ajoodha. Music genre classification: A comparative study between deep learning and traditional machine learning approaches. In *Proceedings of 6th International Congress on Information and Communication Technology (ICICT 2022)*, pages 239–247. Springer Singapore, 2022.
- [LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- [LeN19] Alexander LeNail. Nn-svg: Publication-ready neural network architecture schematics. *Journal of Open Source Software*, 4(33):747, 2019.
- [LVADC07] Edith LM Law, Luis Von Ahn, Roger B Dannenberg, and Mike Crawford. Tagatune: A game for music and sound annotation. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, volume 3, page 2, 2007.
- [LWM⁺09] Edith Law, Kris West, Michael I Mandel, Mert Bay, and J Stephen Downie. Evaluation of algorithms using games: The case of music tagging. In *Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR 2007)*, pages 387–392, 2009.
- [MBL⁺19] Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. Layer-wise relevance propagation: an overview. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 193–209, 2019.
- [MLB⁺17] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification

- decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.
- [MW21] Sakib Mostafa and Fang-Xiang Wu. Chapter 3 - diagnosis of autism spectrum disorder with convolutional autoencoder and structural mri images. In Ayman S. El-Baz and Jasjit S. Suri, editors, *Neural Engineering Techniques for Autism Spectrum Disorder*, pages 23–38. Academic Press, 2021.
- [PG20] Nikki Pelchat and Craig M. Gelowitz. Neural network music genre classification. *Canadian Journal of Electrical and Computer Engineering*, 43(3):170–173, 2020.
- [PPNCP⁺18] Jordi Pons Puig, Oriol Nieto Caballero, Matthew Prockup, Erik M Schmidt, Andreas F Ehmann, and Xavier Serra. End-to-end learning for music audio tagging at scale. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR 2018)*, pages 637–644, 2018.
- [RA22] Sindu Rajendran and SP Anandaraj. Implementation of computationally efficient and accurate music auto tagging. In *Proceedings of the 8th International Conference on Advanced Computing and Communication Systems (ICACCS 2022)*, volume 1, pages 1630–1635. IEEE, 2022.
- [RRS21] Srividya Tirunellai Rajamani, Kumar Rajamani, and Björn W Schuller. Towards an efficient deep learning model for emotion and theme recognition in music. In *Proceedings of IEEE 23rd International Workshop on Multimedia Signal Processing (MMSP 2021)*, pages 1–5. IEEE, 2021.
- [SCDS77] A. Carlisle Scott, William J. Clancey, Randall Davis, and Edward H. Shortliffe. Explanation capabilities of production-based consultation systems. *American Journal of Computational Linguistics*, pages 1–50, February 1977. Microfiche 62.
- [SPM⁺22] Mitt Shah, Nandit Pujara, Kaushil Mangaroliya, Lata Gohil, Tarjni Vyas, and Sheshang Degadwala. Music genre classification using deep learning. In *Proceedings of the 6th International Conference on Computing Methodologies and Communication (ICCMC 2022)*, pages 974–978, 2022.
- [Swa81] William R. Swartout. Explaining and justifying expert consulting programs. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, volume 2 of *IJCAI’81*, page 815–823. Morgan Kaufmann Publishers Inc., 1981.
- [SZ15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*, 2015.

- [TBTL08] Douglas Turnbull, Luke Barrington, David Torres, and Gert Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech and Language Processing*, 16(2):467–476, February 2008.
- [TGT18] Etienne Thuillier, Hannes Gamper, and Ivan J Tashev. Spatial audio feature discovery with convolutional neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2018)*, pages 6797–6801. IEEE, 2018.
- [TKT10] D. Tingle, Y.E. Kim, and D. Turnbull. Exploring automatic music annotation with acoustically-objective tags. In *Proceedings of the 11th ACM International Conference on Multimedia Information Retrieval (MIR 2010)*, pages 55–62. ACM, 2010.
- [TL19] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*, pages 6105–6114. PMLR, 2019.
- [Vat20] Igor Vatolkin. Evolutionary approximation of instrumental texture in polyphonic audio recordings. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2020)*, pages 1–8. IEEE, 2020.
- [WCNS20] Minz Won, Sanghyuk Chun, Oriol Nieto, and Xavier Serra. Data-driven harmonic filters for audio representation learning. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2020)*, pages 536–540. IEEE, 2020.
- [WFBS20] Minz Won, Andrés Ferraro, Dmitry Bogdanov, and Xavier Serra. Evaluation of cnn-based automatic music tagging models. In *Proceedings of the 17th Sound and Music Computing Conference (SMC 2020)*, 2020.
- [WFW19] Bolun Wang, Zhong-Hua Fu, and Hao Wu. Augmented strategy for polyphonic sound event detection. In *Proceedings of the Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC 2019)*, pages 1496–1500. IEEE, 2019.
- [XUD⁺19] Feiyu Xu, Hans Uszkoreit, Yangzhou Du, Wei Fan, Dongyan Zhao, and Jun Zhu. Explainable AI: A brief survey on history, research areas, approaches and challenges. In *Proceedings of the 8th CCF International Conference on Natural Language Processing and Chinese Computing (NLPCC 2019)*, pages 563–574. Springer, 2019.
- [YNDT18] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, 9:611–629, 2018.

- [ZGCH21] Jianlong Zhou, Amir H. Gandomi, Fang Chen, and Andreas Holzinger. Evaluating the quality of machine learning explanations: A survey on methods and metrics. *Electronics*, 10(5), 2021.

Cross-Validation Result for Assigning Feature Maps to Descriptors

- A.1 Cross-Validation Result - Convolutional Layer 1**
- A.2 Cross-Validation Result - Convolutional Layer 2**
- A.3 Cross-Validation Result - Convolutional Layer 3**

A. CROSS-VALIDATION RESULT FOR ASSIGNING FEATURE MAPS TO DESCRIPTORS

Table A.1: CV Results - Convolutional Layer 1

Model	metric	avg-score	guitar	string	drums	fast	violin	piano	synthesizer	male vocals	slow	female vocals
KNN-3	F1-score	29.3%	32.9%	28.0%	42.9%	34.7%	31.0%	31.5%	21.5%	20.5%	40.9%	8.8%
	Recall	35.4%	46.0%	23.8%	60.1%	37.5%	28.2%	46.7%	25.6%	13.8%	67.2%	5.0%
KNN-5	F1-score	28.3%	31.8%	26.3%	43.3%	35.4%	29.3%	31.6%	20.3%	18.3%	41.8%	4.9%
	Recall	33.8%	42.0%	20.7%	60.8%	36.2%	25.0%	46.3%	21.7%	11.5%	70.8%	2.7%
KNN-7	F1-score	27.5%	30.5%	24.5%	44.5%	35.4%	27.3%	31.4%	18.8%	16.5%	42.5%	3.3%
	Recall	32.5%	37.5%	18.5%	62.1%	35.0%	23.0%	44.9%	18.3%	10.1%	73.6%	1.9%
RF-500	F1-score	16.8%	26.5%	12.5%	21.7%	19.4%	7.9%	29.6%	2.5%	19.4%	20.7%	7.7%
	Recall	10.0%	16.5%	7.0%	13.1%	11.3%	4.4%	18.0%	1.3%	11.7%	12.2%	4.1%
RF-200	F1-score	16.0%	25.1%	11.9%	20.8%	19.3%	6.4%	30.1%	1.1%	18.8%	21.9%	4.6%
	Recall	9.5%	15.5%	6.5%	12.6%	10.9%	3.3%	18.2%	0.7%	11.3%	13.1%	2.5%
RF-100	F1-score	14.4%	23.4%	10.4%	18.0%	17.7%	4.5%	29.2%	0.4%	16.7%	21.7%	1.8%
	Recall	8.4%	14.1%	5.5%	10.6%	10.1%	2.2%	17.9%	0.1%	9.9%	13.1%	1.0%

A.3. Cross-Validation Result - Convolutional Layer 3

Table A.2: CV Results - Convolutional Layer 2

Model	metric	avg-score	guitar	string	drums	fast	violin	piano	synthesizer	male vocals	slow	female vocals
KNN-3	F1-score	27.2%	33.8%	27.3%	35.7%	33.1%	25.6%	23.6%	19.5%	23.8%	38.7%	11.1%
	Recall	33.1%	53.5%	28.1%	41.8%	42.7%	25.1%	23.7%	22.2%	22.5%	63.9%	7.6%
KNN-5	F1-score	26.0%	33.0%	26.3%	34.9%	33.5%	23.8%	22.7%	16.9%	22.5%	39.7%	6.9%
	Recall	30.7%	51.0%	25.1%	40.2%	42.3%	21.6%	19.9%	17.1%	20.3%	65.9%	4.0%
KNN-7	F1-score	24.9%	32.1%	25.3%	34.7%	33.3%	22.3%	20.6%	14.7%	21.9%	40.3%	4.2%
	Recall	29.3%	48.9%	23.7%	38.9%	43.7%	19.7%	17.1%	13.9%	18.9%	66.3%	2.3%
RF-500	F1-score	16.7%	19.6%	12.9%	29.6%	23.8%	7.9%	29.8%	1.6%	15.0%	23.9%	2.5%
	Recall	10.2%	11.5%	7.4%	19.7%	14.7%	4.2%	18.3%	0.9%	8.6%	15.5%	1.3%
RF-200	F1-score	16.2%	18.9%	13.5%	28.5%	23.4%	7.4%	30.3%	0.9%	13.9%	23.4%	1.5%
	Recall	9.8%	11.1%	7.7%	18.5%	13.9%	4.0%	18.7%	0.4%	7.8%	14.8%	0.8%
RF-100	F1-score	15.3%	17.4%	13.7%	26.6%	22.3%	7.8%	29.9%	0.3%	13.0%	21.8%	0.5%
	Recall	9.2%	10.0%	7.8%	16.9%	13.3%	4.1%	18.6%	0.0%	7.3%	13.4%	0.1%

A. CROSS-VALIDATION RESULT FOR ASSIGNING FEATURE MAPS TO DESCRIPTORS

Table A.3: CV Results - Convolutional Layer 3

Model	metric	avg-score	guitar	string	drums	fast	violin	piano	synthesizer	male vocals	slow	female vocals
KNN-3	F1-score	27.0%	29.9%	25.3%	35.9%	38.8%	23.2%	25.9%	18.7%	22.7%	35.3%	14.4%
	Recall	37.2%	41.6%	28.6%	59.5%	89.5%	19.7%	22.7%	18.8%	28.2%	42.5%	20.4%
KNN-5	F1-score	25.4%	28.9%	23.2%	35.9%	38.5%	21.5%	25.4%	15.3%	19.5%	35.7%	9.6%
	Recall	33.9%	39.5%	26.2%	58.7%	89.1%	16.2%	20.3%	11.7%	28.0%	41.0%	7.9%
KNN-7	F1-score	23.9%	27.6%	20.9%	35.0%	37.5%	19.5%	24.3%	13.3%	18.5%	35.2%	7.3%
	Recall	29.7%	38.7%	25.1%	33.7%	85.8%	13.9%	18.0%	9.1%	28.0%	39.5%	5.5%
RF-500	F1-score	20.7%	22.1%	18.7%	27.2%	26.9%	14.2%	29.9%	8.7%	19.8%	30.7%	8.9%
	Recall	13.9%	14.8%	12.5%	19.1%	17.1%	9.5%	18.5%	5.9%	12.1%	23.7%	5.7%
RF-200	F1-score	18.1%	20.3%	15.5%	26.3%	26.3%	9.8%	29.7%	2.2%	19.7%	27.7%	3.8%
	Recall	11.2%	12.1%	9.1%	16.9%	16.4%	5.3%	18.3%	1.2%	11.8%	18.7%	1.9%
RF-100	F1-score	16.8%	18.2%	13.1%	24.9%	25.9%	8.2%	29.8%	0.6%	19.0%	26.4%	1.8%
	Recall	10.2%	10.7%	7.4%	15.5%	15.9%	4.5%	18.5%	0.3%	11.1%	17.1%	1.0%