Why Pop? A System to Explain How Deep Learning Models Classify Music

by

Shubham Sharma

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved November 2022 by the
Graduate Supervisory Committee:

Chris Bryan, Chair
Troy McDaniel
Mohamed Sarwat

ARIZONA STATE UNIVERSITY

December 2022

ABSTRACT

The impact of Artificial Intelligence (AI) has increased significantly in daily life. AI is taking big strides towards moving into areas of life that are critical such as healthcare but, also into areas such as entertainment and leisure. Deep neural networks have been pivotal in making all these advancements possible. But, a well-known problem with deep neural networks is the lack of explanations for the choices it makes. To combat this, several methods have been tried in the field of research. One example of this is assigning rankings to the individual features and how influential they are in the decision-making process. In contrast a newer class of methods focuses on Concept Activation Vectors (CAV) which focus on extracting higher-level concepts from the trained model to capture more information as a mixture of several features and not just one. The goal of this thesis is to employ concepts in a novel domain: to explain how a deep learning model uses computer vision to classify music into different genres. Due to the advances in the field of computer vision with deep learning for classification tasks, it is rather a standard practice now to convert an audio clip into corresponding spectrograms and use those spectrograms as image inputs to the deep learning model. Thus, a pre-trained model can classify the spectrogram images (representing songs) into musical genres. The proposed explanation system called "Why Pop?" tries to answer certain questions about the classification process such as what parts of the spectrogram influence the model the most, what concepts were extracted and how are they different for different classes. These explanations aid the user gain insights into the model's learnings, biases, and the decision-making process.

ACKNOWLEDGMENTS

My two years at ASU pursuing a master's degree in computer science have been full of ups and downs. But I can proudly say that I achieved what I came here for, which was learning. This thesis was a major part of that pursuit of knowledge and there are so many people who have supported me in this journey whom I want to thank from the bottom of my heart.

The first one goes to Dr. Chris Bryan. When I came to the US, I was sure I wanted to pursue a thesis track, but the topic of interest was still a blur. Until I took Dr. Bryan's data visualization class, looked at his research, and came across the field of Explainable Artificial Intelligence. It showed a nice mixture of technicality yet scope for creativity and playing around with visualizations. Many thanks to Dr. Bryan for accepting me as one of his thesis students while giving me the freedom of working on exactly what I wanted to. In the lab, he, and his Ph.D. student Jinbin Huang have been very helpful. Their guidance and brainstorming sessions have been crucial for getting any significant research value out of this thesis.

Finally, I would like to thank my family for supporting me in this journey. They have always done more than I ever asked for and I am forever grateful. Last but not the least, I want to thank my friends who have been ever so helpful in this journey. I want to thank them for listening to me when I was stuck even when they had no idea sometimes what I was talking about and more often than not, talking to them have always somehow given me the solutions I was looking for. Just want to thank everyone for bearing with me and accommodating me all this while.

TABLE OF CONTENTS

Page

Page

CHAPTER

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 The Problem Statement

The research question I am focusing on in this thesis is, why are certain decisions made when a deep learning model classifies songs into music genres? To elaborate:

- What influences the model's decisions more or less?
- Are there certain parts of the songs that adversely affect the model's decision?
- If music is converted into images while being fed to the model, how can humans make sense of this incomprehensible way of sensing music?
- Is there a human-friendly way to see what the model has learned?

The problem arises primarily due to the black-box nature of the deep learning models. Further, the model learns by interpreting vectors in very high dimensions which are hard to understand for humans. To exacerbate the problem even further, the state-of-the-art methodologies convert audio into black-and-white graphs/histograms which are then fed to the model to find patterns and classify songs into different categories. What is even baffling is the fact that most of the recommender systems these days make recommendations not based on the similarities between actual songs themselves but rather parse user data and try to match the user's similarities while making recommendations which is far away from what users perceive to be the basis of the recommendations. These problems lead to a lack of understanding of how the model works as well as potential misinterpretations from the user's point of view. Through this thesis, the hope is to alleviate some of

these problems and inspire more work in the field of XAI and especially music considering how big of an industry music streaming is.

1.2 Contributions

This thesis aims to build a system to explain how a deep learning model classifies music into pre-defined genres. The final goal for the system is to help the user identify any patterns that the model might have found to classify similar songs together as well as help the user find anomalies so that they can be handled by possibly fine-tuning the model. The system, which is composed of both backend and frontend, seeks to perform the following tasks for the user:

- Analyze model results from a global level to individual samples.
- Extract human-understandable concepts from the trained model and present them in a visually understandable way. These concepts can be further investigated in terms of genre and concept similarities.
- Be able to present more information on drilling down into a selected group of concepts. Help find patterns among model inputs as well as leverage the use of handcrafted features that are human-understandable.
- Finally, look at an individually selected sample and be able to see the concepts as part of the selected sample while correlating their presence throughout the song being played.

The above tasks when successfully performed will help the user gain insights into the global learning of the model as well as the local explanations of the model for a selected sample.

The implementation of the system was done using python for the backend and react for the frontend. The system can run in a browser and the data for the system is supplied using a flask server. To train the model, the songs were preprocessed to be converted to a dataset of grayscale spectrograms supplied during training using a

data generator. The spectrograms were also augmented to further increase the number of samples available to the model. The deep learning model used for training was resnet34 [14] with imagenet [15] weights. The trained model graph was further used for extracting concepts from the model using concept extraction algorithms. The concepts were further post-processed for aggregation into cluster using k-means clustering technique with the cluster center dimensions reduced to two using dimensionality reduction for plotting on a 2D plot.

The frontend of the system allows the user to perform explorations at a global level, an intermediate level, and a local level. The global level helps the user compare the model's concepts across genres. The intermediate level enables the user to compare concept examples within a selected concept. Using human understandable features, the system also helps the user to verify their conclusions around concept examples. The local level allows the user to explore the concepts found within a selected song sample and follow along by playing the song.

As a result of using the system, the user was able to find good and noisy concepts in the model across genres. The user was also able to explore the concepts further and draw conclusions about why the model finds the concepts important. These conclusions were verified using the human understandable features incorporated in the system. Finally, the user was also able to explore the concepts within a selected song and correlate the important concepts extracted with the selected sample and verify the model's reliability or anomalies.

1.3 Thesis Outline

The thesis is structured in the following manner:

Chapter 2 focuses on the research that has already been done in the field of music information retrieval as well as AI model interpretability. It also highlights the pros and cons of different research methodologies considered for music classification

and feature interpretability, and why some methods were preferred over others for the problem at hand. It also talks about some of the background knowledge required to understand why it makes sense to convert songs into images/spectrograms, what spectrograms are, and then how to use them with state-of-the-art computer vision models to leverage the advances already made in the field.

Chapter 3 describes the dataset for the task. The dataset used for model training is Free Music Archive (FMA) [13]. The dataset consists of 8,000 song clips of 30 seconds each with 1000 songs for each genre. It presents details about the size and limitations of the dataset as well as how the metadata of the dataset is also used to help improve the explainability. It further elaborates on the techniques and pre-processing used to convert audio data into images (spectrograms). This includes using data augmentation techniques such as adding silence randomly in the songs or masking some of the frequency bands before model training.

Chapter 4 talks in detail about the methodology for building the system. It outlines the backend setup for the system, preparing the dataset for the front end of the system. How it pre-processes and consumes the audio dataset, as well as the post-processing, is done after obtaining a trained model that generates data for individual components of the front-end system. The next section lists the components of the front-end system, the use of individual components, key decisions taken while designing the components as well as how the components are interlinked to provide the explanation the system promises.

Chapter 5 discusses two usage scenarios to demonstrate how the system supports explaining the model's behavior and decision making, both from a global

perspective to a drilled-down view of concepts present as well as their influence within a particular sample.

Chapter 6 focuses on the limitations and assumptions of the system as well as simplifications to manage the complexity of the problem.

Chapter 7 concludes the thesis by summarizing the results and reasonings and further concludes with possible future work in the field of XAI as well as research focusing particularly on MIR.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Artificial Intelligence and Deep Learning

One of the simplest definitions of Artificial Intelligence (AI) is the ability of computer systems to perform tasks that otherwise require intelligence to perform [19]. These can be tasks such as perception, reasoning, discovering meanings, learning from examples and experiences, and understanding reasons behind various decision-making processes. While there are various behaviors associated with intelligence, a large amount of research in AI focuses on learning, reasoning, problem-solving, perception, and general use of language to communicate as key goals and components among many others. One subset of AI I will be focusing on in this thesis is Deep Learning (DL) which in turn is a subset of Machine Learning (ML).

Briefly, machine learning or ML can be defined as a field in AI that focuses on the learning aspect of intelligence [20]. It consists of devising algorithms and statistical models that help recognize patterns in mostly large datasets. As a subset of ML, DL uses a particular technique of using artificial neural networks to learn insights from the data. The basic principle behind a neural network is to mimic how a human brain is an interconnection of small individual neurons that work together to make complex decisions or identify complex patterns or solve problems. In the case of deep learning, these artificial neurons are mathematical functions that are connected to many other artificial neurons by the way of taking in inputs and sending out outputs. These inputs and outputs are further manipulated with weights assigned to each neuron. As such, in deep neural networks neurons are organized in layers. Typically, these neural networks are organized in layers, and they learn progressively where initial layers

focus on learning low-level features and the deeper layers focus on learning higher-level "concepts". This progressive method of extracting features is something that differentiates deep neural networks from typical machine learning models where features are already extracted and provided to learn from as part of the dataset the learning is taking place from.

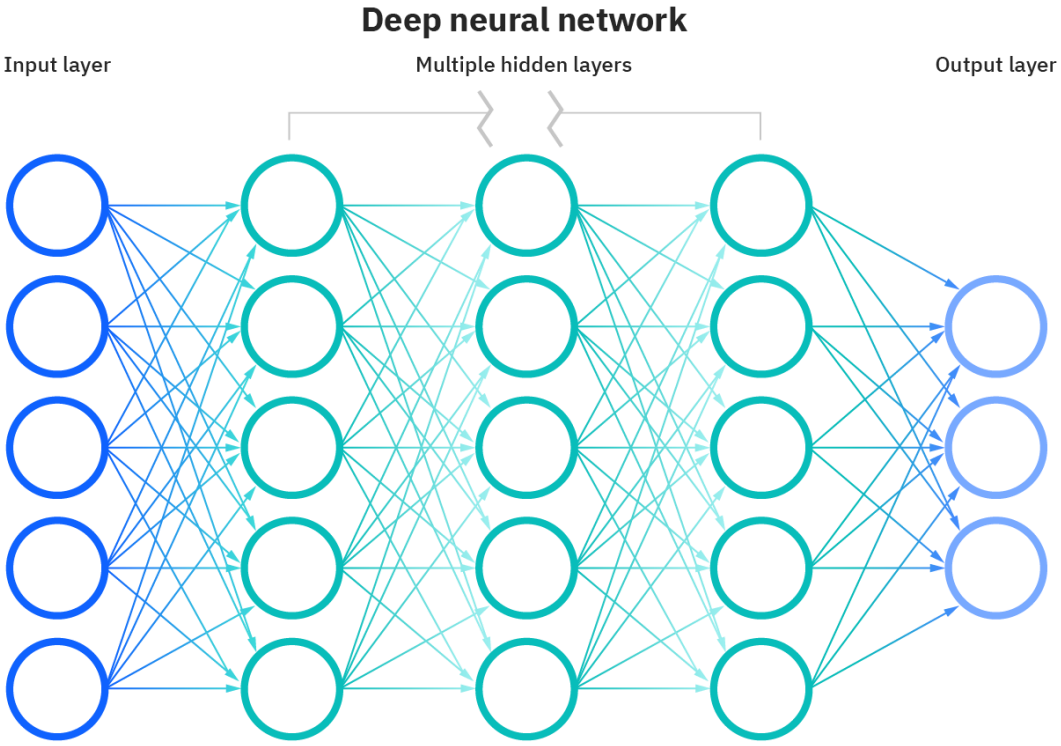

Figure 2.1 A typical multi-layered neural network organization. Source: IBM [21]

2.2 Music Information Retrieval (MIR)

MIR is the science of extracting information from music for different applications. A common task in the field of MIR is music classification into a pre-defined list of genres such as rock, pop, classical, jazz, etc. Other MIR tasks that leverage machine learning are mood classification, artist classification, instrument identification, music tagging, and music generation. Some of these tasks also involve experts who

help extract some pre-requisite information/features before further problems can be solved with the help of machine learning models. Automatic music transcription and content generation are also popular tasks of MIR [22]. An example application of content generation is generating music based on a set of choices or similarities to pre-existing music [23].

2.3 The Need for Explainable Artificial Intelligence (XAI)

XAI, also often known as Interpretable Artificial Intelligence, is the field of AI focusing on making the process of decision-making and the factors influencing those decisions transparent and human-understandable [24]. One of the goals of XAI is increasing transparency in black box ML models. This becomes especially important for high stakes use cases such as in the field of healthcare [25]. Explaining the decisions made by an ML model or the recommendations made by an ML model designed to help humans make complex decisions also establishes trust in the AI and human synergy [27].

Another goal of XAI is to help build ethical AI [28]. With the immense potential and power AI offers, it becomes utterly important to make sure that it is being used ethically. Ethical AI doesn't just focus on following the law but also focuses on making sure that it doesn't make decisions harmful to society [29]. For example, I want to use AI to automate an industrial process to reduce costs. That doesn't mean that AI can make decisions that are harmful to the environment. I can't decide based on the final cost of the process only, I need to make sure that those decisions are ethical too. Similarly, unethical use of AI is seen in spreading misinformation for personal benefits, scamming people [30], or any other kind of fraud. Having decisions explained in cases like these where there may be unintended consequences becomes very crucial.

XAI has also been shown to remove biases in processes [32]. As humans, it is natural to have biases, consciously or unconsciously [33]. At the same time, it is

important to make sure that those biases don't creep into AI-aided systems. Hence, it again becomes important to get explanations for those decisions. For example, an Applicant Tracking System (ATS) is widely used by recruiters and companies to manage candidate applications. If an ATS were to discriminate against a certain subgroup of applicants, it could result in companies missing out on talented candidates, a loss of growth opportunities, a lack of diversity in the company, and damage to the company's public image as well among other things. Biases like these can only be identified if someone can investigate and understand the decision-making process of the automation at use.

Content generation using AI is when AI generates content for the user with very small inputs from the user about the content. This content is primarily for entertainment or to be shared on the internet. With better ML models available these days for content generation, quality is an aspect that is being closely looked at. To make sure AI can be reliably used for automation/content generation, explanations are necessary to make sure that AI understands what makes a unique, yet sensible art. Jukebox [23] is one such application for music generation. Given an artist, genre, and lyrics as input, Jukebox [23] can generate new music. Another content generation application released in the field of Natural Language Processing (NLP) is released by Open AI called GPT-3 [35] which focuses on making the best use of NLP to learn content generation from the internet. One can get this model to write essays, research papers, chat in real-time, and write code based on a short natural language prompt. Another example is DALL·E 2 which can generate realistic and artistic images from descriptions in natural language.

2.4 Current Methodologies

2.4.1 Music Classification

Music classification has long been explored as one of the key classification tasks in MIR [36]. Before diving into music classification specifically, for any classification task including MIR, an important task is feature extraction [1] [37].

2.4.1.1 Feature Extraction

A. Handcrafted Features

One of the earliest efforts for automatic music genre classification was carried out by George Tzanetakis and P. Cook in 2002 [1]. They proposed three sets of features extracted from the audio signal: timbral texture, rhythmic content, and pitch content. Timbral texture focuses on discriminating music from speech using a Short Time Fourier Transform (STFT) calculated for a lot of short time frames of music (usually 256). Another technique relying on Mel Frequency Cepstral Coefficients (MFCC) was used for speech recognition in music. These are very famous techniques widely used in speech recognition problems. More details can be found in [1]. Rhythmic content aims to extract information about the beats of the audio. It builds beat histograms representing information about the strongest beat, sub-beats, and the correlation between different beats while giving more weight to beats like the stronger beats and less weight to weaker beats. Pitch content focuses on providing information about the strongest pitch, sub-pitch, and their correlations.

Another way to extract feature vectors is by combining the feature spaces extracted by binary classifiers [3]. The technique is called space decomposition where a binary classifier is trained to predict a single genre for an audio file and multiple classifiers are trained for the same data sample against different genre samples. The result is a weighted decision of all the classifiers giving out a final label. Two well-

known techniques for training these multiple classifiers are One Against All (OAA) and Round Robin (RR) [3]. The OAA classifier built for a class treats the same class samples as positive samples and other class samples as negative samples. The RR classifier built for a class treats the same class samples as positive, another selected class samples as negative, and the rest of the class samples are discarded. Finally, the feature spaces from all these classifiers are combined in a weighted manner to form a new decomposed feature space. Similarly, features extracted from time segments such as the three mentioned before can be decomposed together as well to form a feature space.

Apart from the handcrafted features mentioned above, there are more features now available courtesy of big streaming services that produce good classification results. They provide features such as tempo, acousticness, speechiness, liveness, danceability, etc. These features are human-understandable and easy to process but, a major drawback of them is that they are time-consuming, hard to automate, and in some cases proprietary.

B. Spectrograms for Model Inputs

Unlike handcrafted features, deep neural networks can automatically extract features but require larger datasets to achieve good accuracy [4, 5, 6, 7]. At the same time, the features extracted via neural networks can be at times high dimensional and difficult for humans to understand.

There is another way to process audio using deep learning models that has produced state-of-the-art results in terms of classification accuracy. This can be achieved by changing the input to the model where each audio file is first pre-processed and converted into a spectrogram to be consumed by the model [8]. The extracted features we have seen can be categorized into two kinds of feature sets. The first feature set focuses on the frequencies, beats, and amplitudes of strong and weak

beats/pitch of the song. The second feature set is extracting useful information from the song due to the variation of the signal with time. Both feature sets can be combined using a spectrogram. A spectrogram is a representation of the frequency spectrum, varying in amplitude across time. It combines the two feature sets by mapping the amplitudes of frequencies and their variations through time in a heat-map-like organization. It essentially does this by creating a lot of small bins in the audio sample corresponding to time and for each of those bins the y scale specifies the frequency and the intensity of a particular point at a particular time corresponding to a particular frequency specifies the amplitude of that frequency at a particular time.

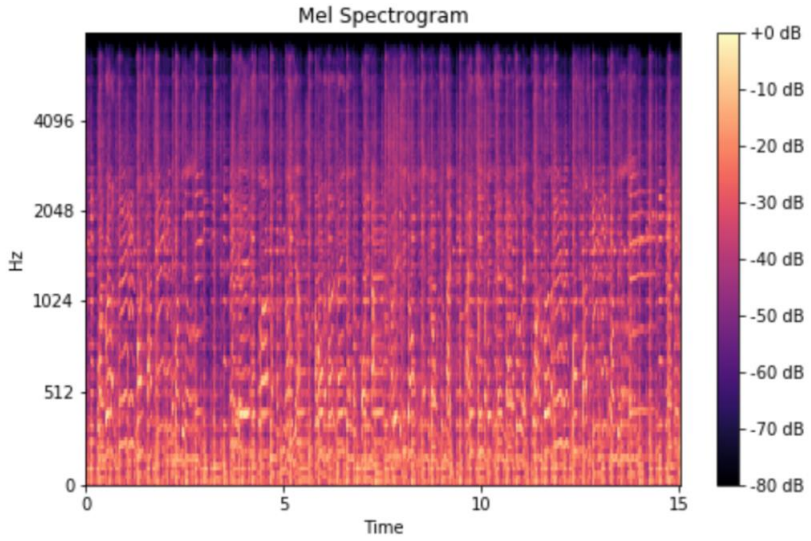

Figure 2.2 A mel spectrogram with mel scale frequency on the y scale and time on the x scale constructed for a 15s audio clip. The scale on the right scales the amplitude for the frequency.

A drawback of spectrograms is that it doesn't account for the fact that humans do not perceive sound linearly. Rather, humans perceive sound logarithmically [38]. For example, it is easier for humans to distinguish between sounds of frequencies 1000Hz and 2000Hz as compared to sounds of frequencies 10000Hz and 11000Hz even though the difference between both sets is of 1000Hz. To tackle this, a mel spectrogram is calculated presented in Figure 2.2, which remaps the frequency scale to the mel

(derived from melody) scale. This ensures that as the pitch of the signal increases, so does the difference between frequencies marking equal distance ticks on the scale. This also ensures that the comparison is done between pitches and not simply linearly increasing frequencies. Similarly, mel spectrograms also use a decibel scale instead of the amplitude to account for how humans perceive loudness as well.

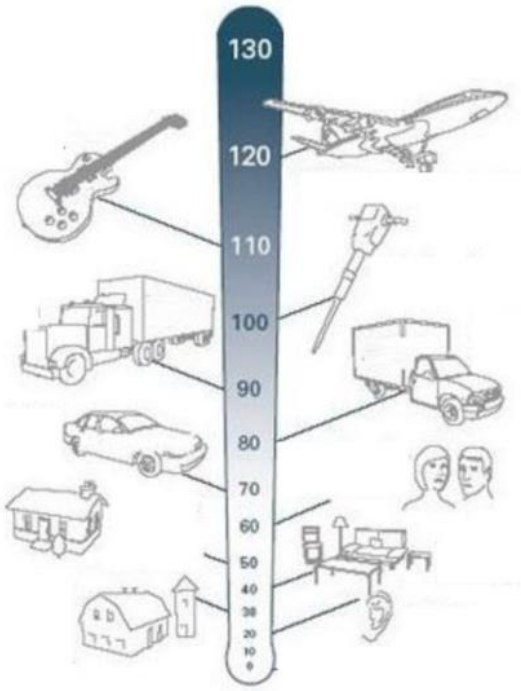

Figure 2.3 Decibel level comparison of different sounds. Source [39]

C. Feature Extraction for Deep Learning

To summarize, there are many benefits for converting audio data to spectrograms. These include:

- It combines both frequency and time domain feature sets into a single feature space.
- It converts an audio signal into an image which can be processed rather easily by the deep learning model.
- It ensures that the input replicates how humans perceive music.

Once these mel spectrograms are fed into the deep learning model, the model can process music just like it is processing images. Hence, it will look for patterns among the images for the purposes of classification and its decisions would consequently be affected by groups of super-pixels swaying the model's decisions towards one genre or the other. Ultimately this converts the problem into one of image classification where the features of the image are represented by the pixels of the image with each pixel row possibly representing one feature vector. Moreover, because it is the intensity of the point in a spectrogram that we care about, spectrogram images can be converted to grayscale images for ease of processing by the model. Hence, each point in the image corresponds to a pixel value ranging from 0 to 255. This methodology is in line with the work done previously [40] that has shown state-of-the-art results on ESC-50 dataset [41] and UrbanSound8k dataset [42] with 92.89% and 87.42% validation accuracies respectively.

2.4.2 Interpretability

As discussed in previous sections, explanations for AI are becoming increasingly important. There have been various ways that researchers have experimented with to help explain a deep learning model. These ways range from presenting basic explanations in a way which are understandable by a layman to using AI in conjunction with an expert allowing them to play around with various parameters.

One of the early efforts into interpretability in images is Saliency Maps [10]. Typically, a map produced using this technique shows how important each pixel of the image was for its classification. This certainly helps reveal relevant regions and provides a quantified measure but, it still focuses on local explanations i.e., a single image.

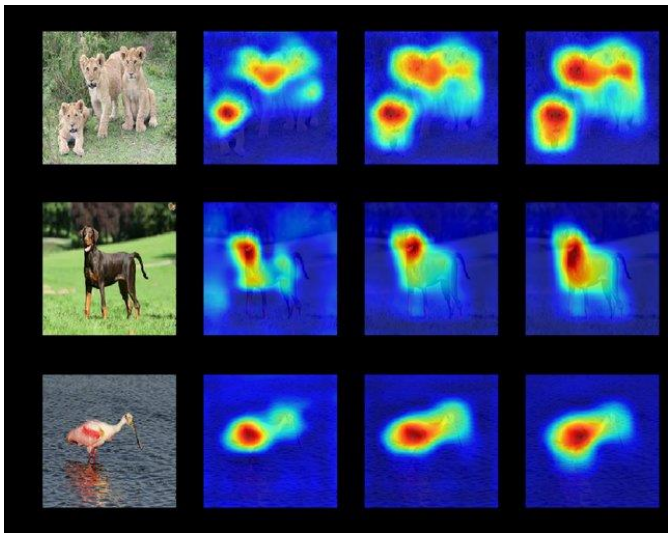

Figure 2.4 A Saliency Map. The closer to red the color, the more important the pixel for the classification of the image. Source [43].

Prototype/Examples based explanations have also been tried aiming to provide explanations to the user by providing examples of classes while highlighting features important for the characterization of the example [11] not just from a single sample point of view but at the class level. This preserves the quantifiable nature of saliency maps and provides a global explanation as well. A drawback of this and similar techniques is that the logic for this needs to be incorporated into the model itself. Hence, it leads to a lack of the ability to be plugged in with any machine learning model.

To tackle this problem, interpretability nowadays is taken care of by post-processing the saved model graphs. Although the extracted features by the deep learning model are high dimensional, they can be traced back to a group of super-pixels/patches called concepts within the spectrogram image. These concepts are more human-friendly, can be visualized easily and a group of concepts extracted using techniques like Testing with Concept Activation Vectors (TCAV) [9] and presented side by side can give a human user a fair idea of the parts of the spectrogram a model

prioritizes for recognizing a particular class. TCAV scores can also work with multiple types of features and prediction classes.

TCAV [9] has been shown to be a successful technique for explainable AI, but it doesn't automatically extract those concepts for the image and evaluate them. Instead the user supplies a set of candidate images that contain the concept. In this research, to tackle this problem Automatic Concept Extraction (ACE) [12] is employed. ACE works by dividing the image into patches of images and evaluating them with the model. Patches more relevant to a category will in general yield better results and can be classified as concepts. Similar patches are then grouped as examples of the same concepts. Finally, concept importance is measured with TCAV scores, and the topmost important concepts are returned.

Traditionally, concepts have been applied to image datasets, where extracted patches are easy to interpret as they represent human-understandable concepts (see Figure 2.5). In this thesis, we apply concepts to spectrograms, which do not have features that can be easily understood by humans. This represents a major research challenge for this thesis, which we discuss in chapters four and five.

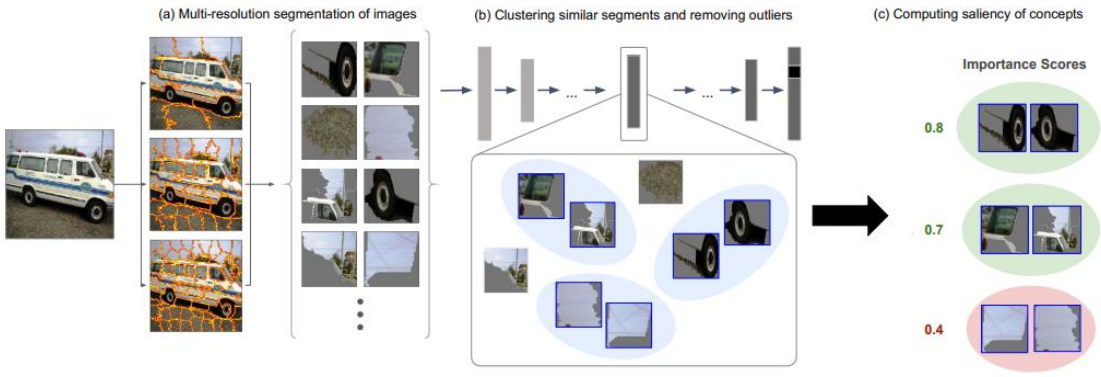

Figure 2.5 ACE process flow. First, a set of patches is extracted. Then they are evaluated against a pre-selected model layer and grouped. Importance scores for groups are calculated and meaningful concepts are returned.

CHAPTER 3

MUSIC DATASET

The dataset chosen for this research is the Free Music Archive dataset (FMA) [13]. The chosen dataset is one of the largest datasets that offer high-quality full-length audio files. Overall, the dataset offers 106,574 full songs featuring 16,341 artists and was published in 2017. This dataset is much larger than comparable open music datasets used in research such as GTZAN [1], which by being limited to 1000 songs makes the application of deep learning techniques difficult. Hence, the chosen dataset provides more variety in terms of songs and hence offers a larger scope for the model to learn.

For this thesis, a subset of FMA offered as FMA small is chosen to demonstrate the tasks. The subset consists of 8,000 high-quality song samples each of 30 seconds in length. The songs are also partitioned into sets of 1000 songs per genre (eight genres in total) which ensures the balance of the dataset and doesn't skew the model training to either side. While each song has a top genre assigned to it, each song also has a list of genres attached to it since there is no standardized method to classify each song into a particular genre in the music industry. The eight top genres are:

- Electronic
- Experimental
- Folk
- Hip-Hop
- Instrumental
- International
- Pop
- Rock

From a data management point of view, a CSV about tracks is made available where each row corresponds to a single track with a unique track id. Other columns of the row have data such as the name of the song, the artist, the top genre, all the genres, play counts, comments, duration, album of the song, location of the song, and many more. All tracks are mp3 format tracks and most of them have a sampling rate of 44,100 Hz, a bit rate of 320 kbit/s (highest quality mp3 bitrate), and in stereo. There are also 518 features in total available that are extracted using the librosa python library. These 518 features include 74 distinct features and for each feature 7 different statistics were computed.

The dataset also offers a set of features calculated by Echonest (later acquired by Spotify). The offered features namely are:

- Acousticness
- Danceability
- Energy
- Instrumentalness
- Liveness
- Speechiness
- Tempo
- Valence

These features further help increase user understanding and readability and the explainability system helps connect the features from the spectrograms with their human understandable counterparts.

CHAPTER 4

WHY-POP: THE INTERFACE

Here, we describe the Why-Pop system that we have built. It consists of two components. One is the machine learning model that forms the backend of the system. The second is the user interface that the user can interact with or use for visualizations for better interpretability of the model i.e., the frontend. Hence the two components of the system are described in the following sections.

4.1 Backend

4.1.1 Data Conversion and Pre-Processing

As discussed in the previous sections, there are several benefits of converting audio data to spectrograms. On similar lines, the FMA dataset was first converted to spectrograms. First, each audio file was rendered into an image of dimensions 224 x 224, as these dimensions are commonly used. These dimensions are very popular for use with deep learning models. Further, the converted spectrogram image was generated in a greyscale format since the real data is defined by the intensity of the pixel for a particular frequency and time and varying color hue values could potentially perturb model performance in undesired ways.

Although, most of the sound clips were in stereo (2 audio channels instead of one) but, since all the clips were not in stereo, one channel was dropped for the sake of consistency. All the files were resampled as well to a standard 44,100 Hz. For files smaller than 30 seconds, they were padded with silence. And for files longer than 30 seconds, they were trimmed to use the first 30s of the clips. The hyperparameters for creating a mel-spectrogram were as follows:

- Sample rate: The sample rate used was 44,100 Hz as standardized earlier. Meaning it extracts 44,100 samples for each second of audio.
- n_fft: This parameter used the value 1024. That means, while creating Fourier Transformations, the signal was divided into 1024 sub-samples. A Fourier Transformation essentially helps break down the signal at a particular time into its constituent frequencies.
- n_mels: The value used for *n_mels* was 128. This is a widely used value and the default as well that specifies the number of bins in the spectrogram on the x scale where each bin corresponds to a single point in time.
- Hop_length: The *hop_length* value was set to 512. It specifies the difference between consecutively sampled segments for the Fourier Transformation. A good value for the *hop_length* parameter ensures the overlap of adjacent segments and prevents information loss.
- Pad: In a case where the number of Fourier Transformations and *hop_length* exceed the sample data; the rest is padded with 0.

To enhance the dataset and add more generality to the model, each spectrogram was converted into four more spectrograms with augmentation. For 8,000 songs, that resulted in a dataset of 32,000 spectrogram samples increasing the amount of data available to train the model. Out of the four spectrograms, one was the originally retained spectrogram, and rest three were augmented with random horizontal/vertical bands of black pixels (silence). For the augmented spectrograms, one had a randomly placed horizontal band across the width of the spectrogram (or image). Another spectrogram had a vertically placed black band across the length. Finally, one had a band placed both horizontally and vertically. Essentially, the horizontal bands acted as frequency masks and the vertical ones acted as time masks. The masking was limited to a maximum of 10% of the image length/width.

4.1.2 Model Training

To feed the data, a data generator was used. It would randomly pick four tracks and generate four original spectrograms from them. Then augment them to make sixteen spectrograms out of those. The training images from the generator were then shuffled along with the labels and passed on to the model. In case of errors, while creating spectrograms from the audio file, the data sample was replaced with another. That essentially makes the batch size to be sixteen as well.

4.1.2.1 Resnet34

For the deep learning model, ResNet34 [14] was used. ResNet is a deep neural network consisting of 34 layers stacked on top of each other. It is a widely popular and well-known neural network for the purpose of image classification and achieved top state-of-the-art results with its release in 2015. Mostly, to train deep learning models for complex tasks, an easy method is to stack more hidden layers into the network to get better results. But it only works until a limit after which, more layers end up leading to deterioration in performance and accuracy. This problem was alleviated by ResNets by using skip connections. More specifically, every two or three layers, a skip connection was added that would skip the layers in the middle and multiply the output of the source layer to the weights of the target layer directly followed by the addition of the bias. Further, in cases where skip connections connect layers with different dimensions, the input is padded with zeros. This architecture is shown in the architecture diagram below.

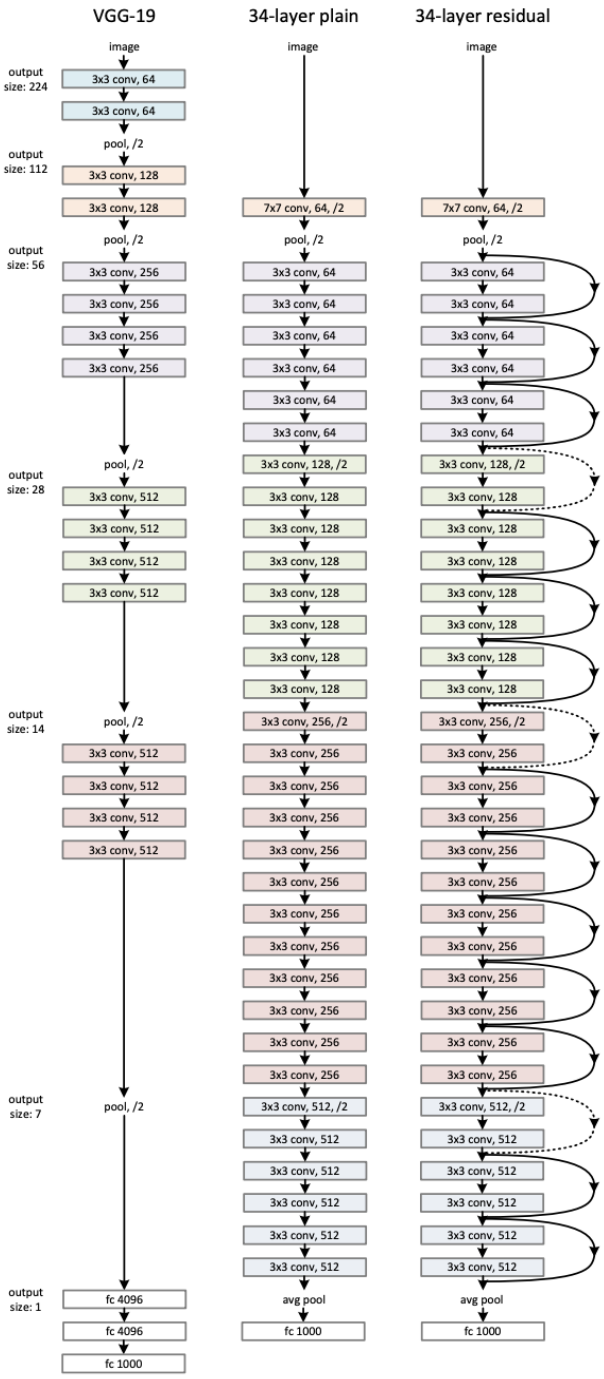

Figure 4.1 Example network architectures for ImageNet [15]. Left: VGG-19 [44] model (19.6 billion Floating-point Operations per second (FLOPs)) as reference. Middle: Plain network with 34 layers (3.6 billion FLOPs). Right: a residual network [14] with 34 parameter layers (3.6 billion FLOPs).

This architecture from ResNet helps get rid of a few common deep-learning problems which are as follows:

- The biggest problem it solves is the vanishing gradient problem [45]. The vanishing gradient problem is faced when training the neural network, where the updates to the weights of the layers of neural networks are propagated by the previous layer as a partial derivative of the error function with respect to the current weights. For deeper neural networks, this becomes a problem as the derivatives turn into very small quantities and hence don't make any major updates to the subsequent layers which ultimately hinders the learning of the model. With skip connections, this is taken care of by ResNets as the gradient doesn't vanish for deeper architectures.
- Another problem ResNets help with is, with an increase in the number of layers, ResNets perform at least as well as their shallower counterparts. That means the layers are good at learning identity functions. In the best-case scenario, this means that the subsequent layers perform better than the previous layers, and worst-case scenario they perform at least as well as the previous layers without any loss of accuracy.

The reasons above lead to better overall performance as well as compared to the plain 34-layer neural network in this case. Finally, ResNet34 comes as part of the PyTorch python library pretrained with final weights after training on the popular ImageNet [15] dataset.

4.1.2.2 Training Loop

To utilize the ImageNet weights and the benefits of transfer learning, the model was only fine-tuned to classify spectrograms (music) into genres. For the training loop, the hyperparameters were as follows:

- Batch Size: The *batch_size* was 16 that contained four different songs along with their augmented spectrograms.
- Epochs: The number of *epochs* for training the model was set to 50 since the aim was just to fine-tune the model.
- Trainable layers: To utilize the model weights from ImageNet, only the fully connected layer and the two immediate convolution layers were set to update weights. Following the idea that layers closer to the end learn more high-level details in images as seen in [9].
- Learning rate: The learning rate was also set to be low at 10^{-4} to preserve the weights of the model from the ImageNet training and utilize transfer learning.

Finally, for each training loop, the weights were updated with the backpropagation technique. The error is calculated for the model outputs compared to the desired output. The error calculated is then propagated backwards to all the previous layers as a partial derivative explained earlier. The loss function used was the cross-entropy loss which has the following formula:

$$-\sum_{c=1}^{M} y_{o,c} \log\left(p_{o,c}\right)$$

Where:

- M – number of classes
- y – 0/1 if the class was predicted incorrectly or correctly
- p – probability if observation o belongs to class c

A key feature of cross-entropy loss is that it penalizes errors heavily when the prediction is wrong, but the model makes the wrong prediction confidently which can be seen in the following graph as well.

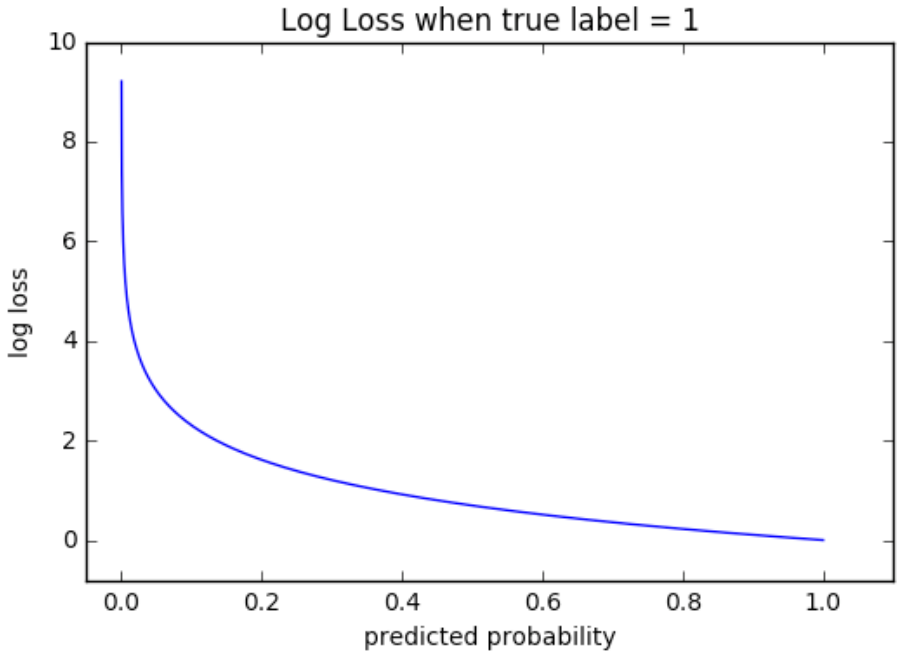

Figure 4.2 Cross-Entropy Loss graph

4.1.3 Concept Extraction

While training the model, the model state and the graph are stored in a state dictionary. This state dictionary is further used by ACE to automatically extract concepts from and evaluate extracted concepts. As discussed earlier, to extract concepts ACE creates image patches from the input image. Since ACE extracts concepts for one class at a time, the input spectrogram images were sorted by classes and then into further subdirectories as required to run ACE. There are in total four hyperparameters that can influence the output from ACE:

- Bottleneck layer: layer4.2.conv2. The second convolution layer in the second block and layer 4 of ResNet34 was selected as the bottleneck layer. This specifies the vector space for concept extraction. Note that this was also the

layer that was trained apart from the fully connected layer during model training.

- num_random_exp: 20. Number of random experiments to perform to evaluate the concepts against.
- num_clusters: 25. This specifies the number of clusters to be extracted with K-Means [17] which directly specifies the number of concepts output.
- n_segments: 10, 15, 25. Number of segments to make with image segmentation techniques to identify important patches from a given spectrogram. The segmentation technique used was SLIC [16].

Since ACE does not support custom models, a few methods within the ACE system had to be implemented as well. The key ones include a method to fetch the activation of the image patches in the bottleneck layer and a method to get the gradient of the fetched activations for the class concepts are being extracted for. As an output for concepts, the concept patches along with the marking for part of the spectrogram they were extracted from, and the TCAV scores of the concepts were returned. As a pre-processing step for the front-end, the concepts were further clustered into 10 clusters using *k*-means clustering based on the Concept Activation Vector (CAV) obtained by getting the mean of the CAV for a concept versus the random experiment datasets which were 20 in this case.

4.2 Frontend

The frontend of the system is built fully in ReactJS. This helped build a website-like interface for the user to interact with and explore the results and extracted concepts from the model, their relations among each other, and get a global to a local level explanation for the model's predictions. The interface is divided into 5 panels and

the interactions lead the user from a global explanation to a more local explanation of the classification of the song. A full view of the interface is as follows:

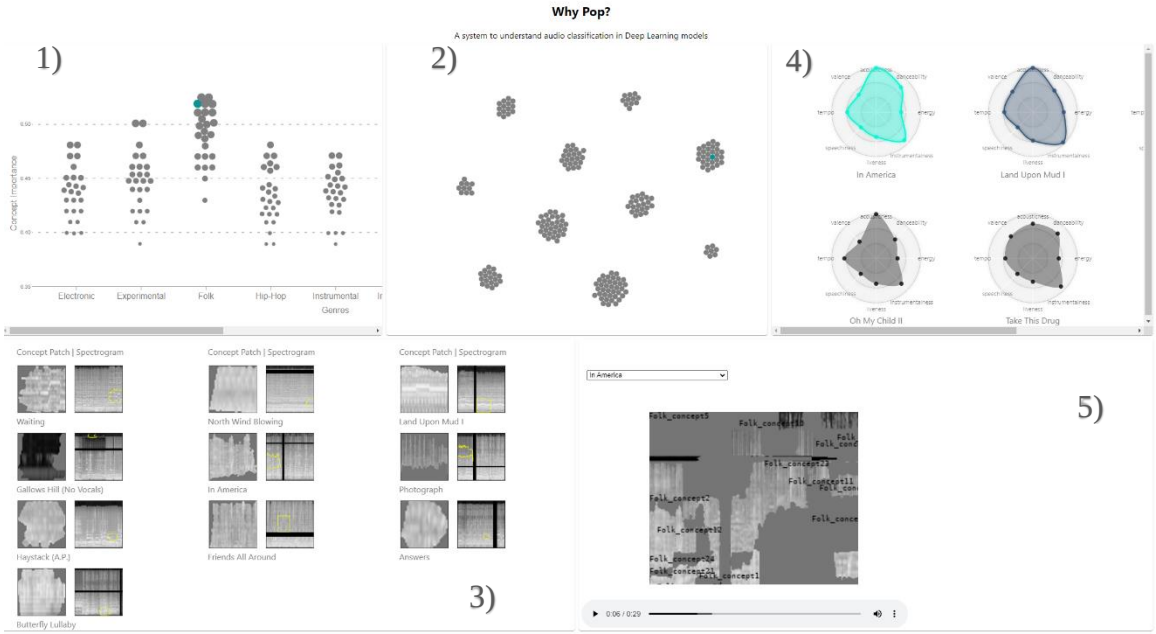

Figure 4.3 Why pop? -The interface, full view. Panel 1-5 after making selections.

4.2.1 Panel 1 – Concepts Importance by Genre

The first panel presents all the extracted concepts across all the genres mapped according to their TCAV [9] (importance) score. This global view helps the user understand how important each concept was in identifying a particular genre. The information is presented in a beeswarm graph with the dots representing concepts and groups of concepts gravitating towards a point on the x scale corresponding to the correct genre. The y scale represents the TCAV score or the concept importance score. The user can use this to identify concepts most important for the genre by hovering over the dots and then clicking to select one of the concepts to present more information about the concept in panel 3. Hovering over a concept not only gives concept information but also highlights the concepts across the genres that have a

similar concept activation vector as well the cluster the current concept is part of in panel 2. In terms of global explanations, this gives the user insights into what important concepts from each of the different genres might be similar. For example, Electronic_Concept17 with the highest TCAV score of 0.5 among all the electronic genre concepts is very similar to a lot of Hip-Hop and Rock genre concepts. An example of these interactions at work is in the Figure 4.4 below.

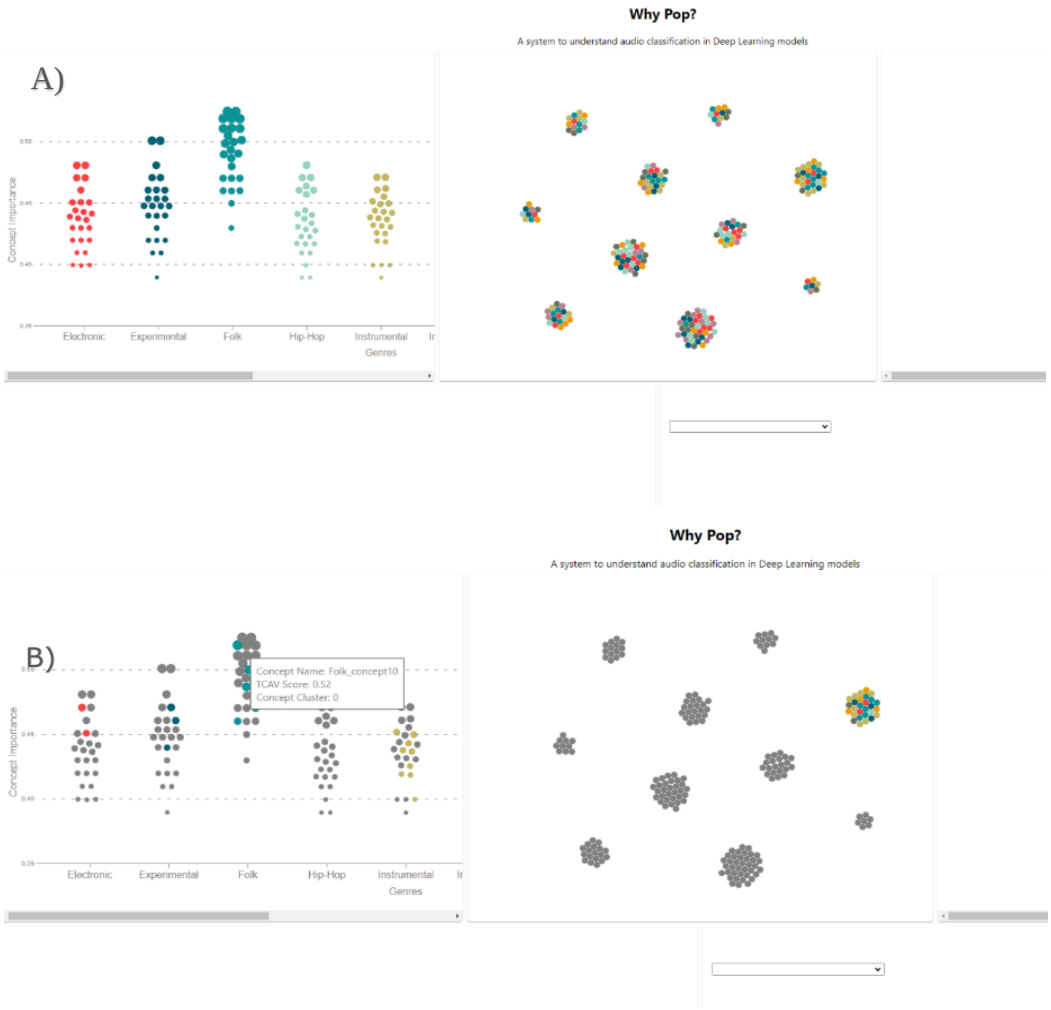

Figure 4.4 A) The first look when the interface loads up. Only Panel 1 and Panel 2 are loaded by default presenting a global view of concepts. B) On hovering over a concept, all the similar concepts are highlighted across the genre and the corresponding cluster is highlighted as well.

4.2.2 Panel 2 – Concept Clusters

Panel 2 of the interface highlights the concepts as part of a cluster in terms of the similarity of the concept activation vectors against all the random experiments which were averaged to form a single concept activation vector per concept. After clustering the concepts into 10 clusters as explained in the backend section, the concept centers were extracted. These high-dimensional concept centers went through a dimensionality reduction technique to be plotted in a 2D space as in panel 2.

The dimensionality reduction technique used was t-Distributed Stochastic Neighbor Embedding (t-SNE) [18]. T-SNE is specifically useful for such use-cases where the aim is to visualize high-dimensional vectors of a neural network being used on a 2D plane to better understand the learnings of that particular layer. Due to the high dimensionality of the dataset/extracted features they are not always linearly separable which is taken care of by t-SNE and hence that is another reason to go for t-SNE rather than Principal Component Analysis (PCA) for example. Since the dataset or extracted vectors were already clustered using *k*-means, this clustering needed to be visible while plotting the vectors in the 2D plane. Also, since the distance between the clusters doesn't mean much in the t-SNE plot, only the concept cluster centers were used to plot the data in the t-SNE plot while clustering the same cluster concepts together. Further, the color for each of the dots representing concepts was preserved from panel 1 and hence represented the mix of genres present in a single cluster. Just like panel 1, hovering and clicking interacted the same way for panel 2 as well where clicking a concept selected that concept for the rest of the exploration. A view of panel 2 can be seen in Figure 4.3 as clusters of concepts. This panel also helps the user get a more global explanation of the model's learnings.

4.2.3 Panel 3 – Concept Images Panel

Panel 3, the concept images panel is populated once a concept has been selected from panels 1 or 2. This panel shows the actual concept patches extracted using ACE and the corresponding spectrograms they were extracted from along with the markings around the parts of the spectrograms they were extracted from. ACE extracts concepts from a set of 50 different examples from the target class. For each concept, it extracts concepts and the corresponding marking in the spectrogram for 10 different songs. In panel 3, those ten songs are presented to the user to help them understand the general areas the concepts are being extracted from and if there are any patterns within the concept extracted. Hence, it gives more intermediate-level explanations between global and local within the concept or the genre. An example could be a concept extracting patches from the higher end of the spectrum in terms of frequencies could be a sign that this concept focuses on guitars. Further, if it has a good TCAV score, it could mean that guitars are an important concept for that particular genre.

Similar insights can be further verified by referring to panel 4 as explained in the following section. An example of panel 3 is shown in Figure 4.4.

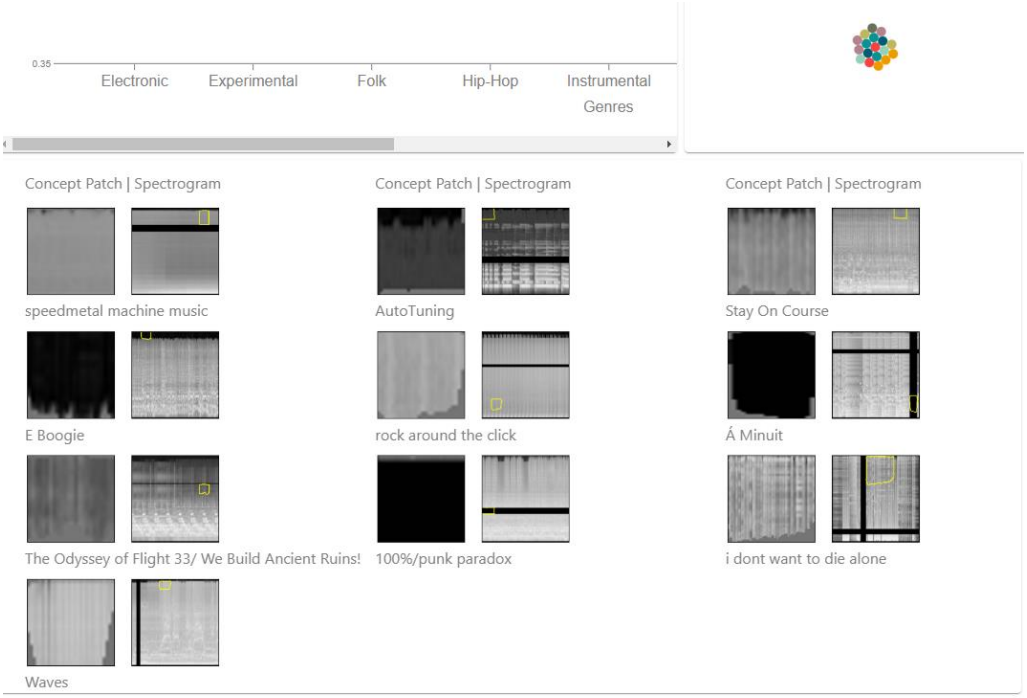

Figure 4.5 Concept Images Panel – shows the extracted concept patches and the corresponding spectrograms.

4.2.4 Panel 4 – Human Understandable Features

Panel 4 just like panel 3 also focuses on intermediate-level explanations. To make sense of the spectrograms in panel 3, this panel uses human understandable features discussed in section 2.4. One thing to note here is though, that these features are not available for all the songs present in panel 3. Hence, only some of the songs are common in both panels but, it still helps give an idea about the patterns in the genre overall. For songs that are common in both panels 3 and 4, if most concepts come out from a higher frequency spectrum in panel 3 and at the same time, acousticness is high for those songs in panel 4, that further points more towards guitar being an important concept. There are eight feature measures in total that a song can be measured upon which were defined earlier in the dataset chapter (Acousticness,

Danceability, Energy, Instrumentalness, Liveness, Speechiness, Tempo, and Valence). The presence of all these feature measures in the song are presented in a radar chart. Therefore, another thing to look for is the similarity between radar chart value blobs across the songs. Further confirming the commonality in songs of the same genre.

For interaction, hovering over a radar blob highlights that song while defocusing the rest of the songs. Finally, clicking on a song selects a song to load up in panel 5 to look at individually.

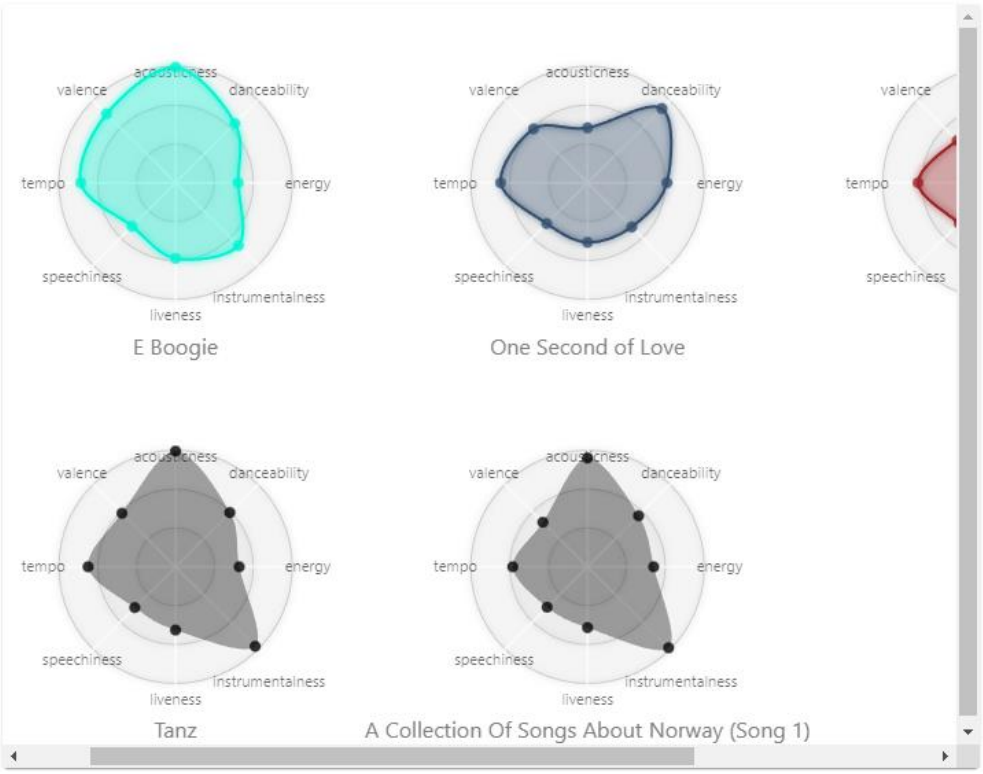

Figure 4.6 Panel 4 - Human Understandable Features, helps the user understand the composition of the song in terms of more human-friendly measures.

4.2.5 Panel 5 – Concepts Within a Song

Finally, panel 5 focuses on local explanations within the song selected from panel 4. This helps the user understand how the concepts are present within the song. Once a song is selected from panel 4, a request is sent to the backend to create a

shape of a spectrogram but with concepts stitched in it while everything else remains grayed out. Since a genre has already been selected, it presents the users the option to choose any song they want out of the 50 different songs ACE extracts concepts for. That also includes the options from either panel 3 or 4 or any other song in the genre the user wants to explore.

To stitch the concepts together, the concept patches were first mapped to their locations in the spectrogram, then overlaid on top of each other, and all the colors apart from the background gray were allowed to override to the front. The locations of the concept patches were used to mark the name of the concept within the spectrogram to get a better idea of the exact concepts present in the song at different points in time. Moreover, this new spectrogram was lined up with an audio player that will allow the user to play the song and identify the beats and pitches of the song corresponding to the concept patch in the new spectrogram.

As mentioned earlier, it gives a more local explanation of how the model processes and learns from the song as well as what were the important parts of the song that the model used to classify the song as belonging to a particular genre. The user can select other songs as well from the dropdown to confirm/reject their hypothesis about a particular genre. An example of this is shown in Figure 4.6.

Figure 4.7 Panel 5, Concepts within a song. Helps the user identify the presence of important concepts at different times within the song.

CHAPTER 5

CASE STUDIES AND RESULTS

The intent with this chapter is to demonstrate how Why-Pop can effectively support tasks related to explaining how our trained neural network predicts the genre of the songs using spectrograms.

5.1 Case Study 1 – Good concepts

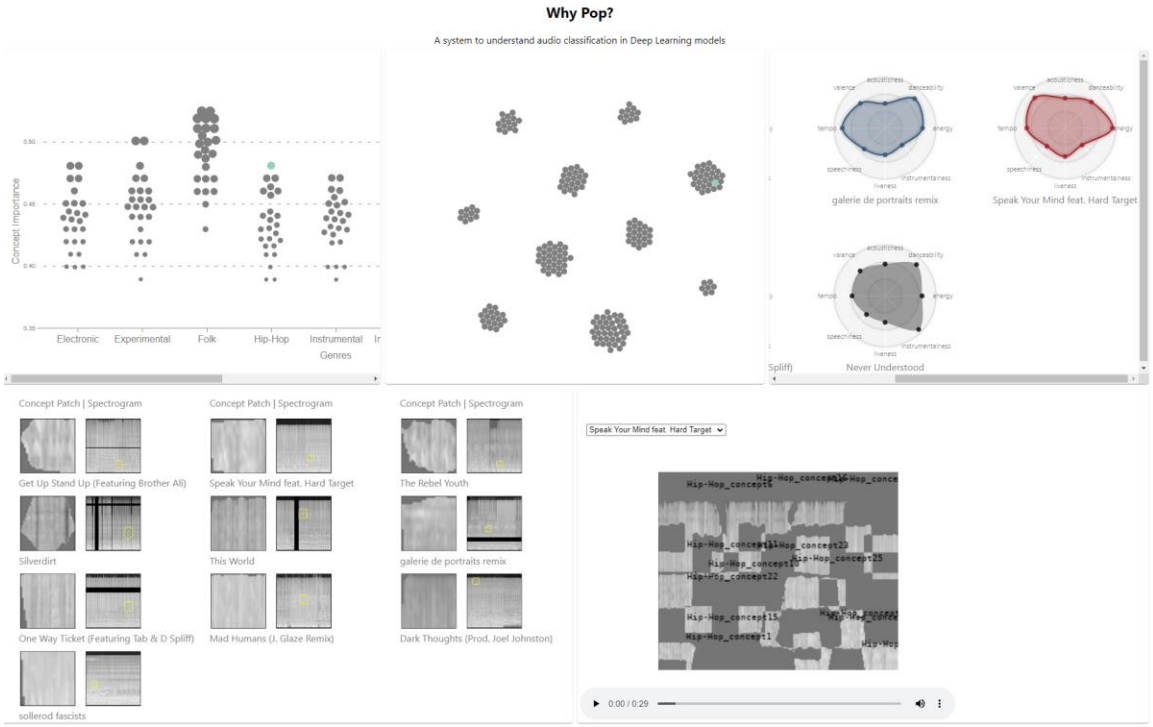

Figure 5.1 This figure illustrates when the example concept, hip-hop concept 18 in case study 1 is explored all the way from global to local view. The panels 1 through 5 are discussed in further detail with zoomed in versions in the following sections.

5.1.1 Global Explanations

First, let's look at "Hip-Hop Concept 18". Looking at the concepts panel, it shows the concept has a TCAV score of 0.48. Considering a range of 0.35 to 0.52 of the TCAV scores for the extracted concepts, that indicates that this is a highly relevant

concept for this class. In fact, it is the highest-rated concept for this class. This score helps explain two things. One is that it helps the user identify that this is a relevant concept for the class and hence should be further explored. The second one is that it also helps quantify the relevance of the concept with the TCAV score which helps with the comparison of different concepts within the class.

Further, the concept clusters panel, panel 2 helps the user find concepts similar to the concept in focus. Looking at both panel 1 and panel 2, we can see that surprisingly it isn't like other hip-hop concepts. Rather, it is similar to a lot of Folk concepts. Similarly, this concept is also similar to a lot of instrumental genre concepts. This can be confirmed in Figure 5.2. These observations help the user gain insights into how the concepts relate within the class and across themselves, serving the task of global explanations through the interface.

Figure 5.2 Circled in red is the concept in focus, Hip-Hop Concept 18. Circled in yellow are the genres containing a lot of concepts like the hip-hop concept but from a different genre. Circled in green is the cluster of concepts irrespective of genre.

5.1.2 Intermediate-Level Explanations

5.1.2.1 Concepts from Spectrograms

Selecting this concept in panel 1, populates panel 3, the concept images panel. This panel will help the user get insights into the concept patches extracted from spectrogram images of the songs as part of this concept. The panel presents 10 examples of concept patches in a table-like manner. In each cell, a single image consists of two parts. The left side of the image shows the extracted patch for the concept, and the right side of the image shows the part of the spectrogram the patch was extracted from. This is marked with a yellow boundary within the image This panel is shown in Figure 5.3. A closer look at the figure shows that most of the concept patches came from the middle or the lower end of the frequency spectrum. This goes on to show that the model considers high bass drum beats an important concept for hip-hop songs. Not just the drums, in fact, the way most hip-hop songs are sung, they usually contain deep voices quite suitable for rap (rhythm and poetry) kind of songs.

It should be noted that ACE randomly chooses examples to extract concepts from and further a subset of 10 is chosen for which this concept patch to concept mapping is provided. Hence, not all the songs can be covered with panel 3.

5.1.2.2 Human Understandable Features

Let’s move on to panel 4 which is also populated as soon as a concept is selected in panel 1 or 2. This visualization can be used in conjunction with panel 3, the concept images panel to further understand what the model has learned and given importance to. The radar charts in this panel break down the songs into 8 human-friendly feature measures as mentioned earlier. Further, it also presents these charts for as many common songs from the concept images panel for which the feature data is available. Nevertheless, this does help the user get insights into the general trends of the songs

across the genre. An example of the songs for which feature data is available is shown in Figure 5.3.

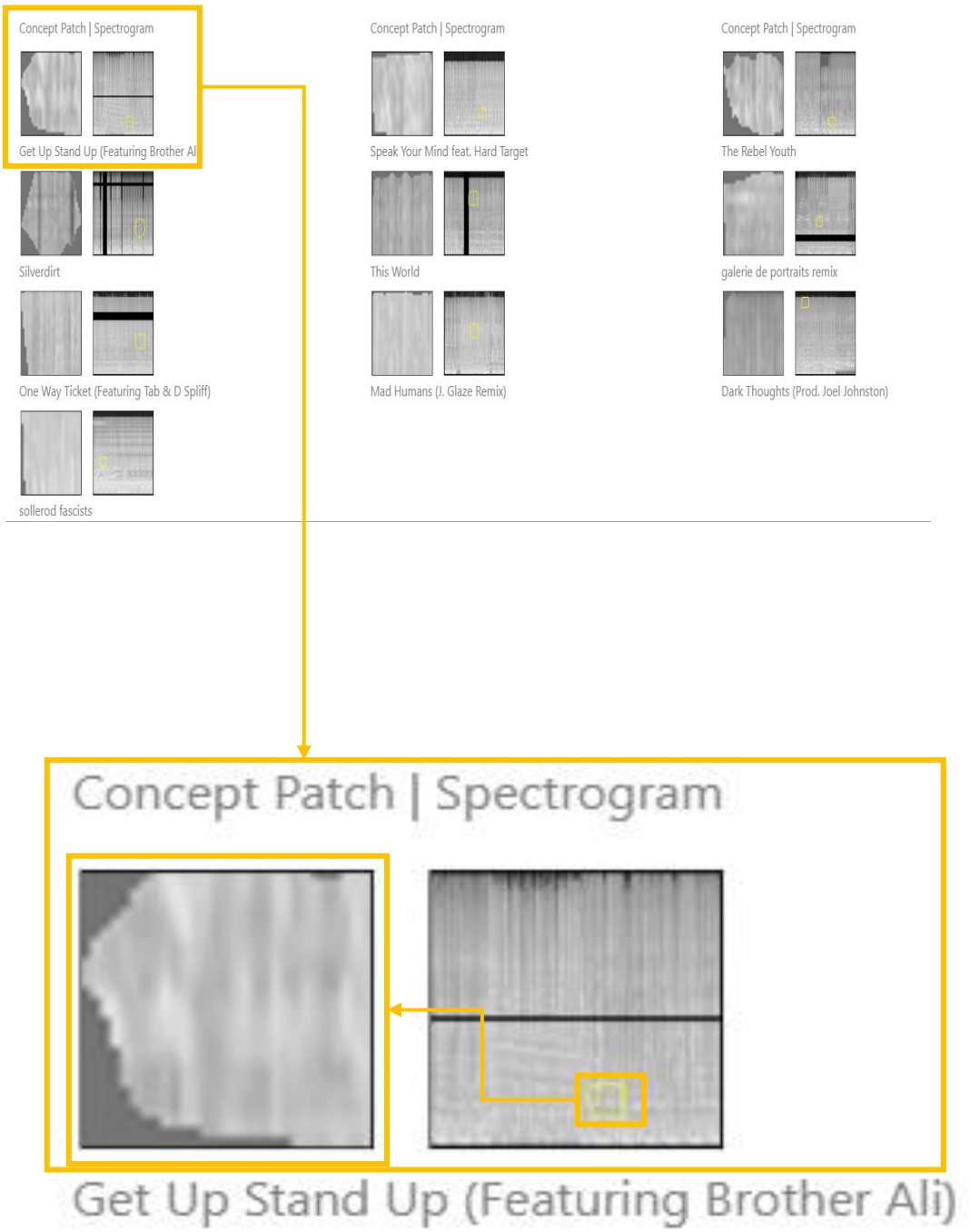

Figure 5.3 Panel 3, Concept Images Panel. Presents concept patches extracted from song spectrogram for hip-hop concept 18. Showing patches extracted from the lower end of the frequency spectrum.

Going back to our observations from panel 3, it was noted that a lot of the concept patches came from the lower end and middle of the frequency spectrum indicating low-pitch sounds. This can be confirmed by the radar charts in Figure 5.4 as in all of the songs acousticness levels are very low. This is in contrast to the songs from another genre such as experimental. Now, tempo indicates rhythm and valence is an indicator cheerfulness of the song. Consequently, tempo and valence levels are generally high across all the songs. The same goes for energy as well as we know hip-hop songs are full of energy. Since most hip-hop songs are composed of rap music as well as a consistent rhythm, that is also in agreement with high tempo and valence values.

On a similar note, hip-hop songs make good dance numbers and hence it is shown with high danceability values. Liveness is an indicator of if the music was recorded live mostly accounting for audience noise. These values are also low noting that these are probably studio-recorded songs. A high value of instrumentalness represents a higher chance of the song being just an instrumental and accordingly, the instrumentalness values are low for these songs. Finally, a value of speechiness above 0.66 indicates the track is just made of spoken words, values between 0.33 and 0.66 indicates the presence of words and music and a value less than 0.33 indicates the song doesn't have any speech. Speechiness values between the 1st and 2nd circle indicate the values are between 0.33 and 0.66 meaning the presence of both music and spoken words. Another thing to note is that the shape of the radar chart blobs of all the songs is pretty similar visually as well indicating similarities across the songs.

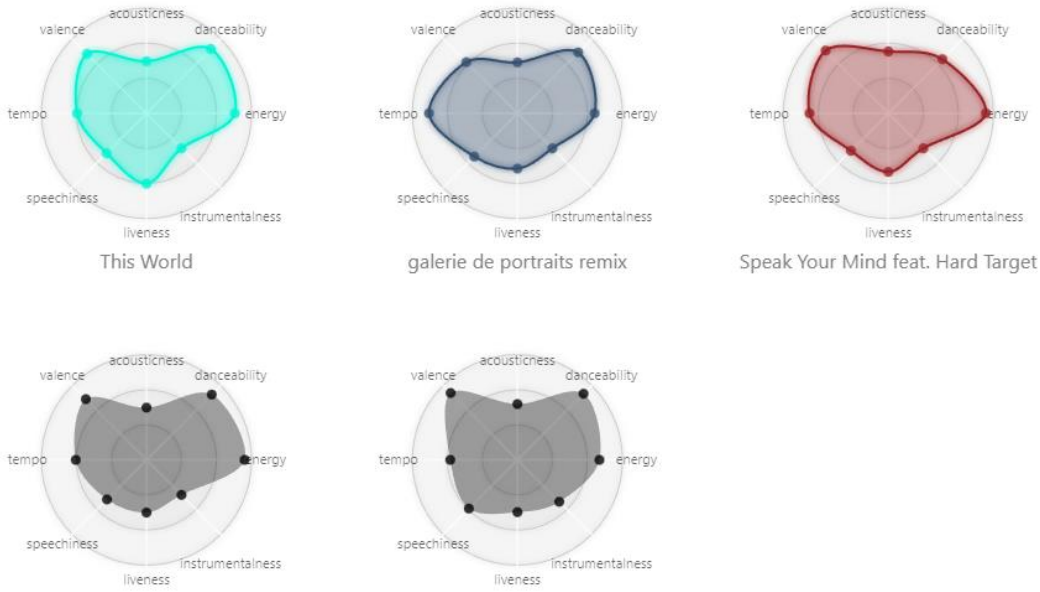

Figure 5.4 Panel 4, Human Understandable Features. Hip-hop songs are compared using human-friendly features.

5.1.3 Local Explanations

Panel 5 focuses on concepts within a selected song. Here, a total of 50 songs are available to be inspected which are the same 50 songs randomly selected by ACE for concept extraction. While any of the 50 songs can be selected for inspection, the user can click on one of the radar charts in panel 4 to select a song and inspect it. For this case study, the song in the top right, "Speak Your Mind feat. Hard Target" is chosen. Once selected, panel 5 presents a spectrogram with only the concept patches present within the song i.e., the group of pixels the model deemed important. This is presented with an option for the user to play the song as well with the seek bar of the player aligned in such a manner that the user can map the current time in the song with the spectrogram and essentially look at the concepts present, their position in accordance to the frequency spectrum and notice the different instruments or vocals probably contributing to one of the concepts.

As an example, in Figure 5.5, the selected song can be played. In line with the name of the song "Speak your mind", the song is very energetic as was seen in panel 4 as well. Good drumbeats and consistent rhythm also lead to high danceability, tempo, and valence values. Something unique in this song is a choir-like high-pitch singing in the background throughout the clip. This is probably accounted for by concepts 6 and 16 present (marked in red in the figure) at the top. But, going back to panel 1, the concepts panel, to check the importance of these two concepts shows these concepts are low in importance for the hip-hop genre which is expected. On the contrary, concepts 15 and 1 present (marked in green in the figure) on the lower end of the spectrogram when looked up in the concepts panel have high importance scores. In fact, concepts 15 and 1 are the 2nd and 4th most important concepts for the hip-hop genre which is in line with the observations and patterns we have come across for the hip-hop genre throughout this case study and expect the model to focus on these concepts while classifying songs for hip-hop genre. This view helps serve the task of local explanations that the user can look at in terms of a single song within the genre and check their hypothesis and gain a better understanding of the model's learnings.

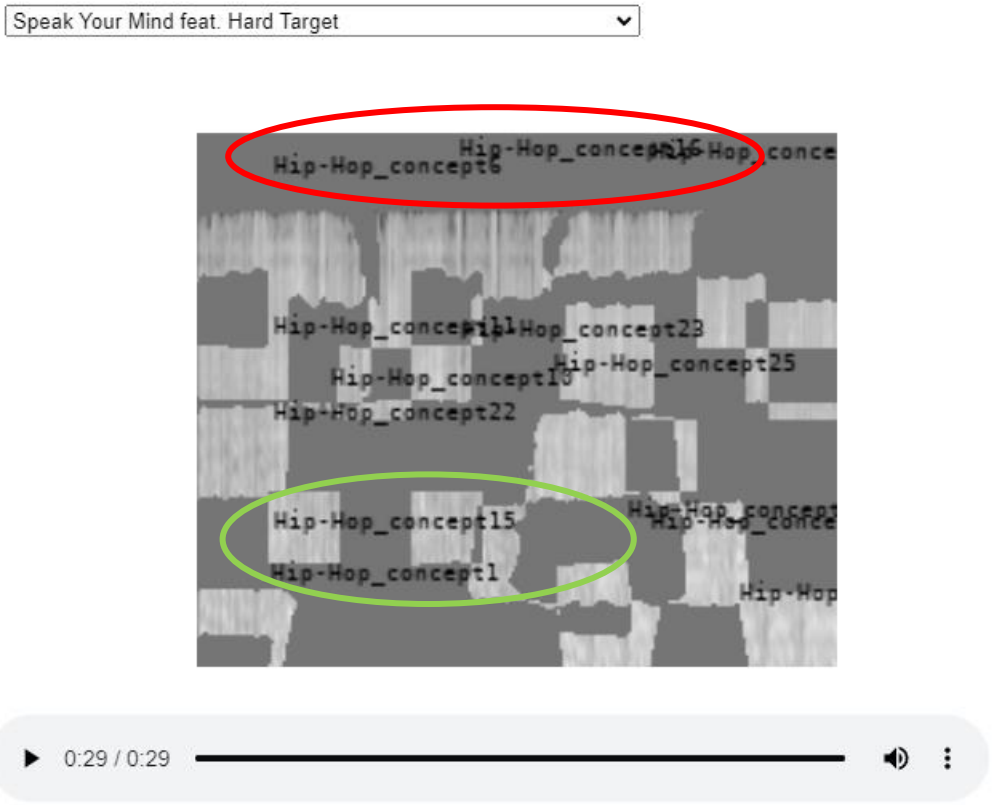

Figure 5.5 Panel 5, concepts present within the song. Looking at the seek bar, the user can map the position back to the spectrogram

5.2 Case Study 2 – Bad Concepts

5.2.1 Global Explanations

For this example, let's look at a concept that the model thinks is very important but, is very spurious and hard to make sense of. Let's look at Folk concept 18 as an example of a bad concept while comparing it with Folk concept 21 as an example of a good concept for reference.

To start off, we can see that while one concept is good and one is bad, both have a TCAV score of 0.52 which is the highest across the range in terms of concept importance for the class. Further, looking at the concept clusters panel as well it can

be seen that the bad concept is part of a very small cluster and when those other concepts are inspected, they have low TCAV scores for their respective genre. For the good concept though, it's part of a big healthy cluster. This can be seen in Figure 5.6 as well. This helps the user uncover the presence of noisy concepts being learned by the model and indicates that they may need to make some adjustments for better results.

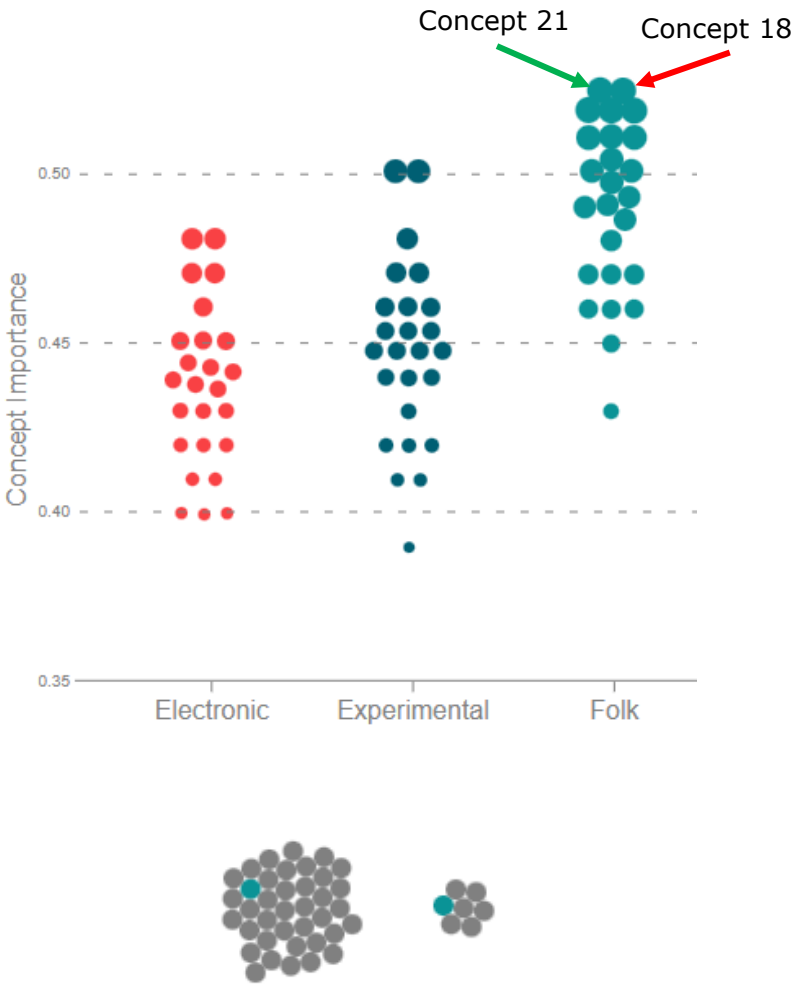

Figure 5.6 Panel 1, Concepts Panel. Good concept marked with green and bad concept marked with red. Panel 2, concepts cluster panel. The good concept cluster is presented on the left and the bad concept cluster is shown on right.

5.2.2 Intermediate-Level Explanations

5.2.2.1 Concepts from Spectrograms

Now, let's inspect this noisy concept further to understand what the model might be picking up on from the spectrogram.

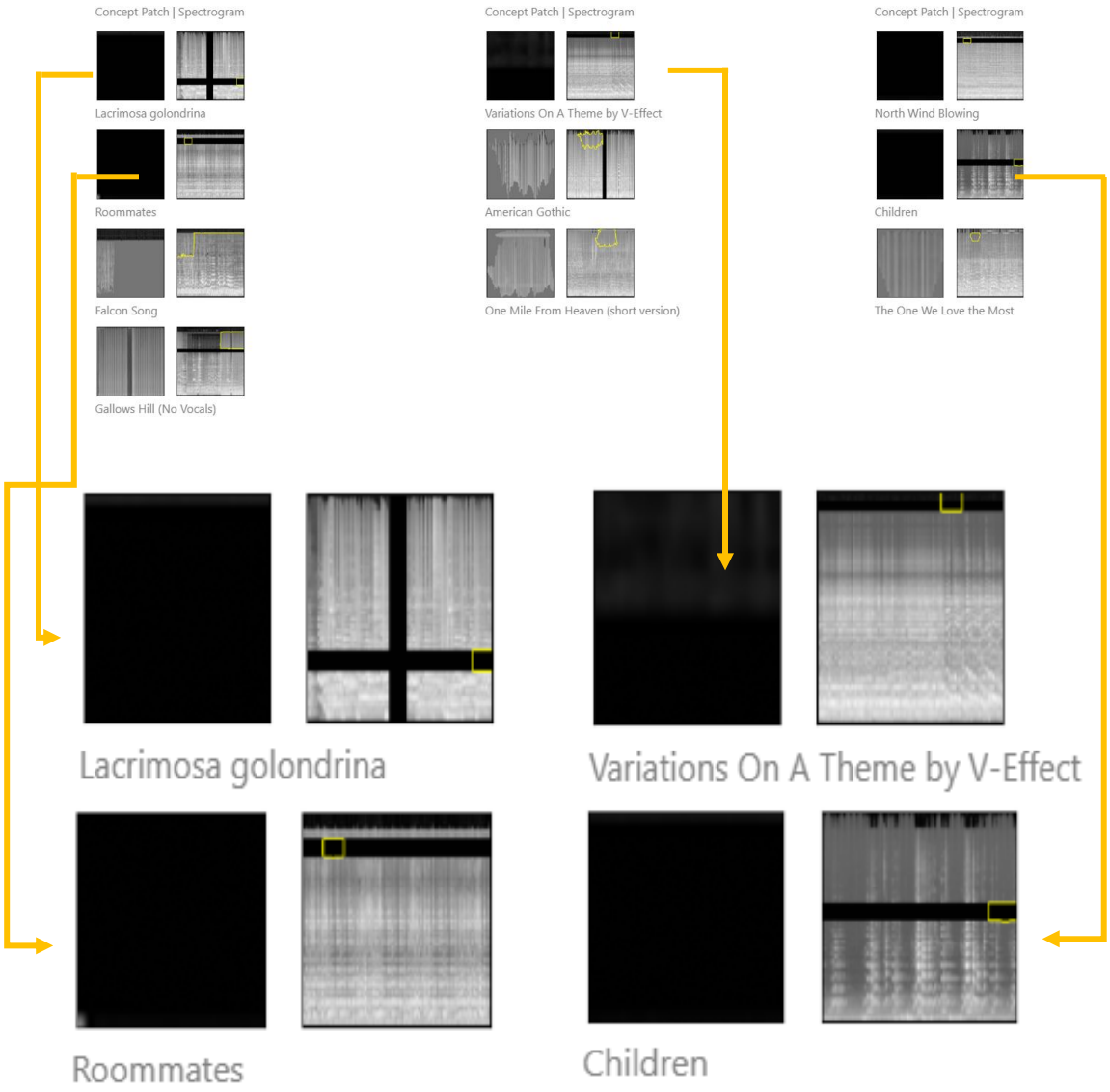

Figure 5.7 Panel 3, Concept Images panel. Below the 10 spectrograms is a zoomed-in view of bad concepts extracted from some of the spectrograms.

A quick look at figure 5.7, the concept images panel tells us that out of the 10 examples, for 5 songs, the extracted concept patches are the black bands that were added as data augmentation. This clearly shouldn't be a concept that the model should be learning, hence helping us uncover these kinds of biases in some cases. Looking at good folk concepts in Figure 5.8 the model seems to be extracting comparatively more meaningful concepts for the class.

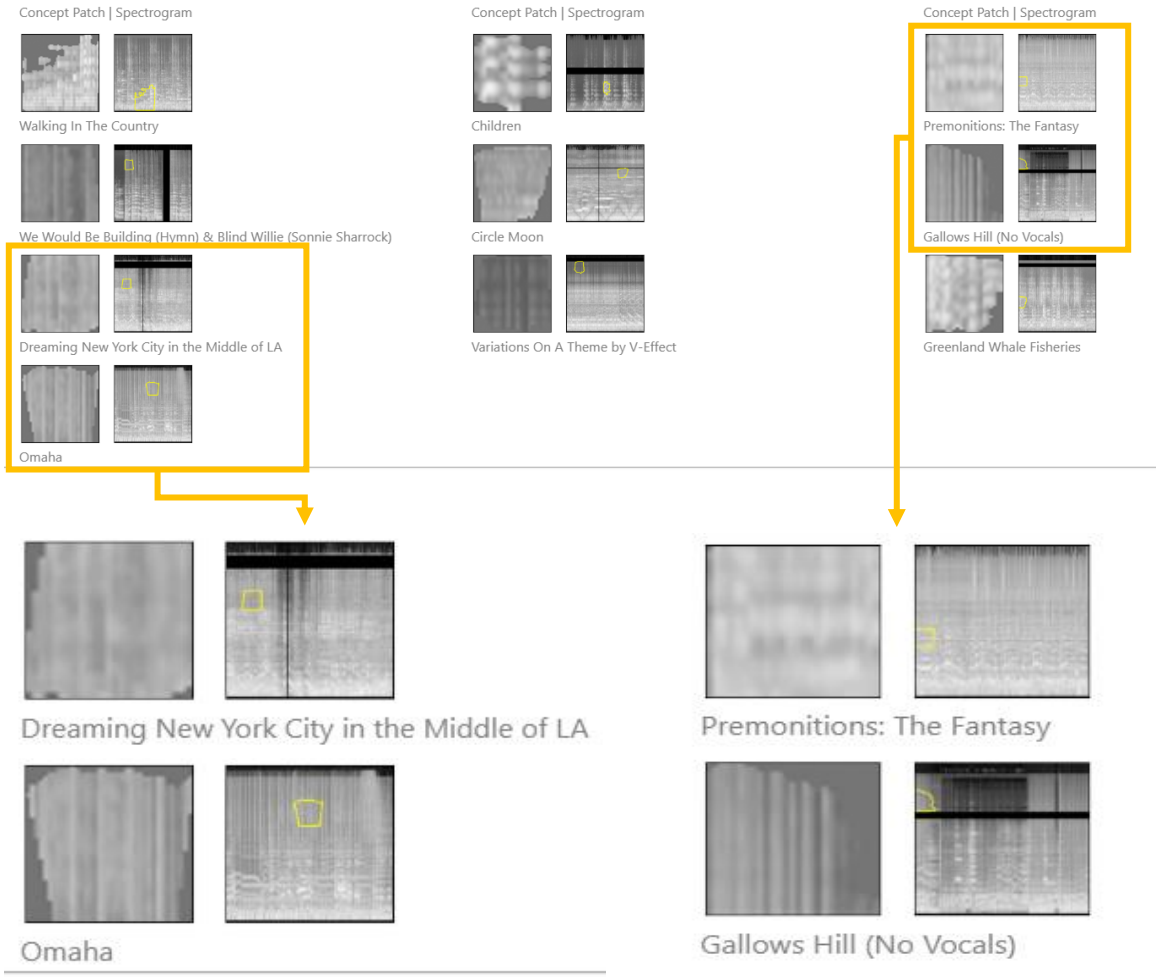

Figure 5.8 Panel 3, Concept images panel. Zoomed-in version shows concept patches extracted from the good folk concept.

The concept patches extracted in the good concept seem to be coming from the higher end of the spectrogram. This is in line with more use of acoustic instruments in the songs. This whole comparison helps the user understand that while the model is able to find good concepts for the class, it is also not reliable against adversaries in the data or song samples, especially in the case of this class.

5.2.2.2 Human Understandable Features

To confirm some of the observations earlier, let's now look at some radar charts with more human-friendly features and try to see if some patterns can be found there. Ideally, for the bad concept, it should be more random compared to a good concept which should reveal some patterns that the model is learning. Comparing the songs from the bad concept and the good concept as shown in Figure 5.9, in terms of similarity, it can still be seen that almost all the songs present high acousticness values which are very trademarks of a folk song. But at the same time if we look for any visual patterns, for the bad concept, only two songs look like having similar values and shapes. While in the case of the good concept, almost 4 out of 5 songs show very similar characteristics. Further strengthening our understanding of the model and the mistakes it might be making while learning from data.

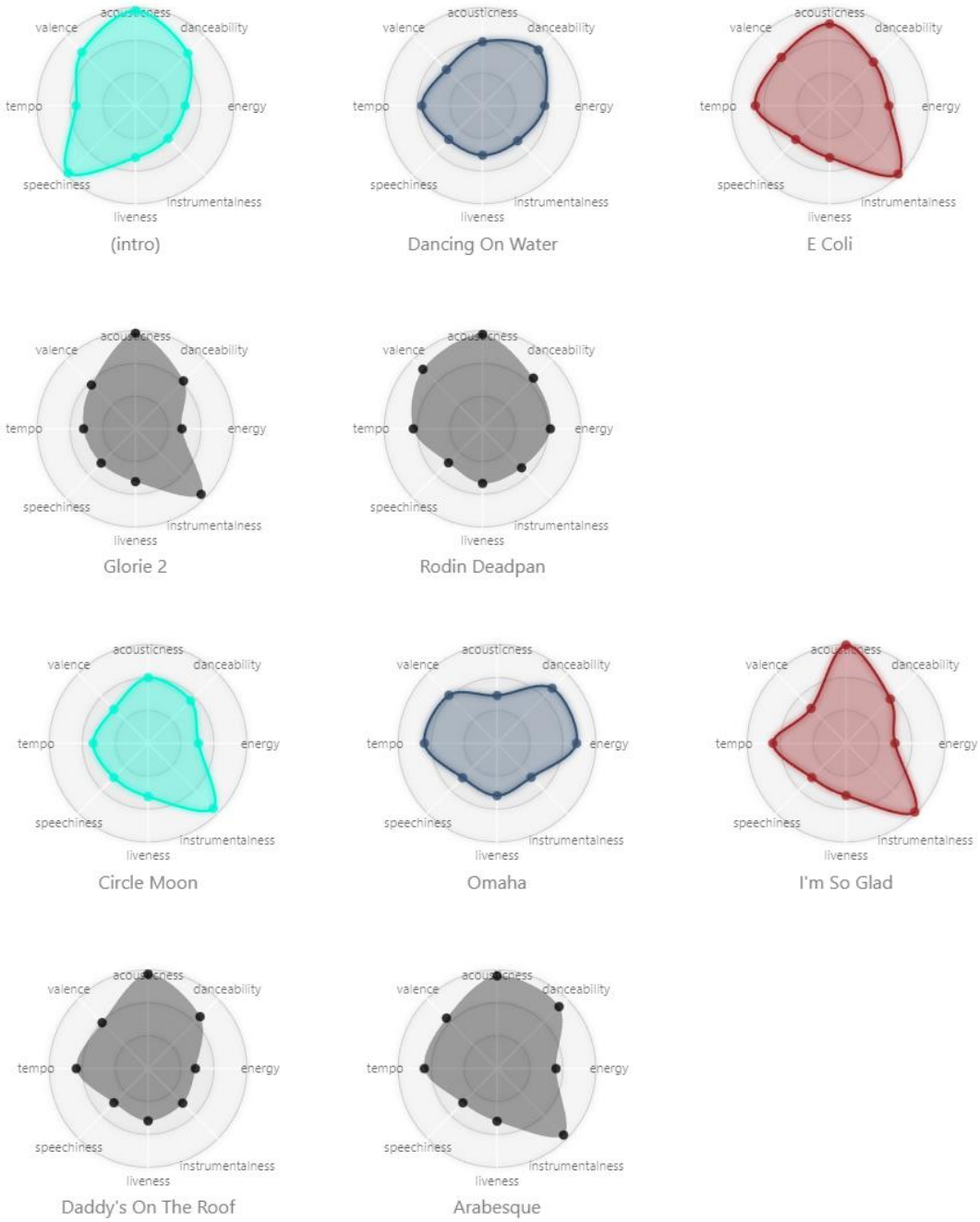

Figure 5.9 Panel 4, Human understandable features. The top 5 songs are songs from the bad concept and the bottom 5 songs are from the good concept

5.2.3 Local Explanations

Finally, selecting one song each from the bad and the good concept batch gives us more insights into comparing the concept examples locally, sample by sample.

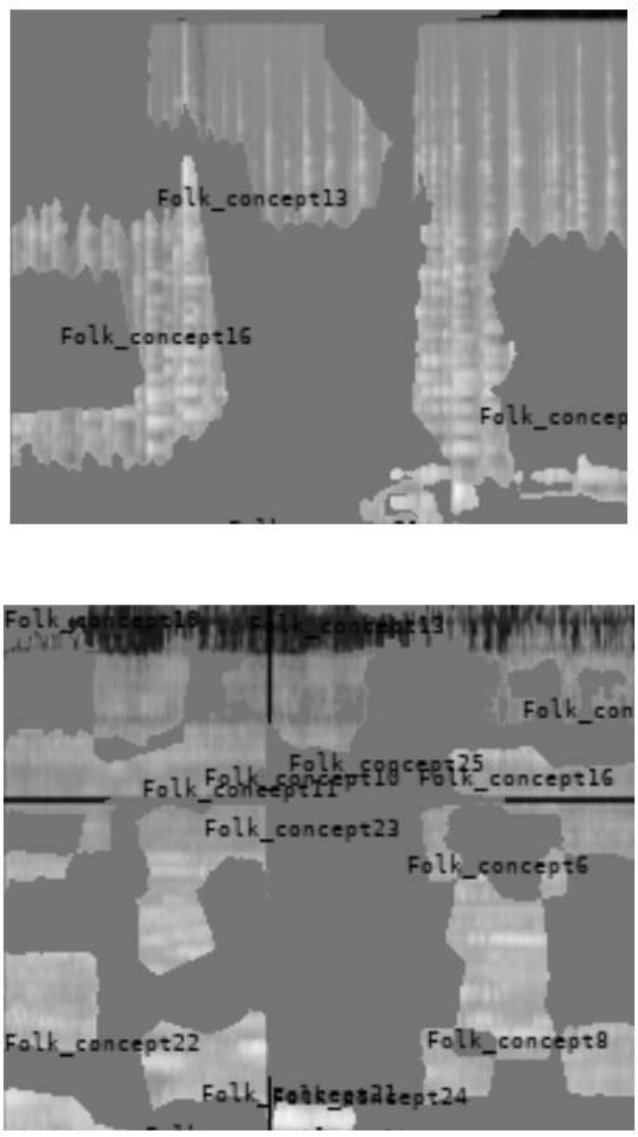

Figure 5.10 Panel 5, Concepts within a song. All concept patches extracted from a single song are stitched together. The song on the top belonged to a bad concept example, the bottom one belonged to a good concept.

Looking at the examples in Figure 5.10 ties all our observations down. For the song representing the bad concept, the model was only able to extract three concepts from it, and that too very large and generic in quality. The model is not able to identify the defining characteristics of the class in this concept at least. Although, if we look at the song from the good concept example, there are lots of concept patches present that probably helped the model classify these songs. It should also be noted that most of the concept patches come from the middle and higher end of the frequency spectrum, agreeing with the high acousticness value we saw earlier. This one-on-one comparison of examples makes it even more certain that despite the high importance score of the bad concept, it should be discarded. The local explanations in this case helped conclude the doubts over the model's learnings in this case.

CHAPTER 6

LIMITATIONS

As with any system, there are certain limitations to this system as well. Starting with the backend, the network itself is dependent on a lot of hyperparameters. Since there is transfer learning involved, for different datasets and networks, the layers we train might reveal different results. Previous research [9] has shown that the layers close to the logit layer have more influence on the prediction but, the number of layers that need to be fine-tuned can always yield different results.

Apart from the model training, the backend is dependent on the concept extraction process. ACE, the algorithm used for concept extraction randomly chooses examples from the target class to extract concepts from which could directly influence the quality of the concepts extracted. The TCAV score that quantifies the concept also uses randomly selected examples to test the concepts against which could again influence concept quality. The clustering of extracted concepts using *k*-means was also based on the hyperparameter *k*.

The system is also dependent on the human understandable features dataset to tie the insights down but, the available dataset was in fact far smaller than the actual dataset which limits the available options the user can drill down into in terms of dataset samples. The availability of this dataset is dependent on either experts handcrafting these datasets or proprietary data made available by streaming companies.

In terms of the frontend, hyperparameters are involved in the creation of spectrograms which can affect the model training, the concept extraction process, and when displayed to the user, their ability to find patterns visually among them. To take care of this though, standard procedures from the industry have been used for their

extraction process. For panel 2, the concepts cluster panel, t-SNE was used which again involves setting parameters such as perplexity which can drastically change how the clusters are presented to the user. The distance and tightness in the clusters are highly dependent on this parameter. To avoid this, K-Means was first used to form the clusters. For highly correlated and non-linear datasets, t-SNE might reveal different results for different perplexity values. Also, since it is an iterative process, the initialization is random and hence can have different results for the same hyperparameter values.

Finally, the biggest limitation is the availability of royalty-free datasets such as the one used here. This becomes especially difficult because deep learning neural networks need a lot of data to learn from. To account for this, data augmentation techniques were used but they still cannot replace the availability of quality data.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 Conclusion

The thesis aimed to explain the model's decision-making process in three levels of granularity: global explanations, intermediate-level explanations, and local explanations. Another goal was to dive deeper into the task of music classification which is thought of as a rather unintuitive process and is often overlooked in comparison to other fields. At the same time, the lack of available data doesn't help to make the process any easier. Despite this, the backend of the interface proposed is nicely able to utilize the research already done in computer vision. Moreover, the explanation part of the interface helps to develop some intuition in the mind of the user.

As seen in chapter 5, the system can help uncover the model's learnings and reveal surprising insights into the concepts model gives importance to. The thesis focused on all three levels of granularity in detail while giving the user the option to explore different data samples as they please. The interface helped reveal patterns within the genre and define aspects of the song the model deemed important for the classification task. In terms of concepts, the interface helped develop an understanding of what the concepts meant and at the same time verify the observations using the human understandable features. Finally, concepts presented within the song along with the audio played definitely helped make important conclusions for the genre/concept the user was looking into.

Finally, the good concept and the bad concept example in chapter 5 was able to ensure that users don't fall into the trap of confirmation bias. Finding anomalies and biases is an important part of the process of creating better and more general models.

One of the goals of XAI is to uncover the unintended consequences of the process. This is exactly what was seen in the case of the bad concept where the model was focusing on black bands in the spectrogram i.e., the silence added as part of data augmentation to make the model more generic. But the examples seen in bad concept were not an intended or an expected consequence of that. Another XAI goal was to establish trust in the user. By using this interface, the user can learn about areas the model is able to excel at and at the same time learn about areas where the model's learnings are poor. This helps the user correct the model's mistakes and over time trust the model to perform reliably.

7.2 Future Works

Hopefully, this work will be able to drive more research into music classification and XAI. At the same time, there is a scope for a lot of improvements on the proposed interface itself. Some of them are as follows:

- Human Understandable Features: The current system relies on the availability of these music features. Enabling the system to automate this would directly extend the comparisons the user can make to understand the model.
- Increased Interactions: Currently, the user can focus on one concept at a time. Scaling that to let the user explore multiple concepts at the same time and side-by-side comparisons would certainly help reduce the number of times the user has to follow the top-down approach.
- Performance: In any software system there is always scope to improve things in terms of performance. To manage the load on the system, a lot of the hyperparameters were chosen for the user with some of the data prepared beforehand. Providing more options to the user while maintaining a high-performance bar will help the user explore the model even better. Along similar

lines, giving the user the option to choose the model used for classification can help them pick the best one for the task.

- User Feedback: While the usability of the interface was demonstrated by using case study examples in chapter 5, another way to test the system's abilities would be through user studies and further note interesting insights the users are able to uncover while incorporating their feedback about the interface to increase usability.

REFERENCES

1. G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," in IEEE Transactions on Speech and Audio Processing, vol. 10, no. 5, pp. 293-302, July 2002, doi: 10.1109/TSA.2002.800560.

2. L. Rabiner and B. H. Juang, Fundamentals of Speech Recognition. Englewood Cliffs, NJ: Prentice-Hall, 1993.

3. Silla, C.N., Koerich, A.L. & Kaestner, C.A.A. A Machine Learning Approach to Automatic Music Genre Classification. J Braz Comp Soc 14, 7–18 (2008). https://doi.org/10.1007/BF03192561

4. E. J. Humphrey, J. P. Bello, and Y. LeCun, "Feature learning and deep architectures: new directions for music informatics," Journal of Intelligent Information Systems, vol. 41, no. 3, pp. 461–481, 2013.

5. T. L. Li, A. B. Chan, and A. Chun, "Automatic musical pattern feature extraction using convolutional neural network," in Proc. Int. Conf. Data Mining and Applications, 2010.

6. P. Zhang, X. Zheng, W. Zhang, S. Li, S. Qian, W. He, S. Zhang, and Z. Wang, "A deep neural network for modeling music," in Proceedings of the 5th ACM on International Conference on Multimedia Retrieval. ACM, 2015, pp. 379–386.

7. A. Alexandridis, E. Chondrodima, G. Paivana, M. Stogiannos, E. Zois, and H. Sarimveis, "Music genre classification using radial basis function networks and particle swarm optimization," in Computer Science and Electronic Engineering Conference (CEEC), 2014 6th. IEEE, 2014, pp. 35–40.

8. Yandre M.G. Costa, Luiz S. Oliveira, Carlos N. Silla, An evaluation of Convolutional Neural Networks for music classification using spectrograms, Applied Soft Computing, Volume 52, 2017, Pages 28-38, ISSN 1568-4946, https://doi.org/10.1016/j.asoc.2016.12.024.

9. Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., & Sayres, R. (2018, June 7). *Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV)*. arXiv.org. Retrieved October 20, 2022, from https://arxiv.org/abs/1711.11279

10. Erhan, Dumitru & Bengio, Y. & Courville, Aaron & Vincent, Pascal. (2009). Visualizing Higher-Layer Features of a Deep Network. Technical Report, Univeristé de Montréal.

11. Kim, B., Rudin, C., & Shah, J. (2015, March 3). *The bayesian case model: A generative approach for case-based reasoning and Prototype Classification*. arXiv.org. Retrieved October 21, 2022, from https://arxiv.org/abs/1503.01161

12. Ghorbani, A., Wexler, J., Zou, J., & Kim, B. (2019, October 8). *Towards automatic concept-based explanations*. arXiv.org. Retrieved October 21, 2022, from https://arxiv.org/abs/1902.03129

13. Defferrard, M., Benzi, K., Vandergheynst, P., & Bresson, X. (2017, September 5). *FMA: A dataset for Music Analysis*. arXiv.org. Retrieved October 22, 2022, from https://doi.org/10.48550/arXiv.1612.01840

14. He, K., Zhang, X., Ren, S., & Sun, J. (2015, December 10). *Deep residual learning for image recognition*. arXiv.org. Retrieved October 23, 2022, from https://arxiv.org/abs/1512.03385

15. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015, January 30). *Imagenet Large Scale Visual Recognition Challenge*. arXiv.org. Retrieved October 23, 2022, from https://arxiv.org/abs/1409.0575

16. Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk, SLIC Superpixels, EPFL Technical Report 149300, June 2010

17. Jin, X., Han, J. (2011). K-Means Clustering. In: Sammut, C., Webb, G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-30164-8_425

18. Van der Maaten, Laurens, and Geoffrey Hinton. "Visualizing data using t-SNE." *Journal of machine learning research* 9, no. 11 (2008)

19. *Artificial Intelligence, N.* artificial intelligence, n. : Oxford English Dictionary. (n.d.). Retrieved November 7, 2022, from https://web.archive.org/web/20220614105900/https://www.oed.com/viewdictionaryentry/Entry/271625

20. Mitchell, Tom (1997). Machine Learning. New York: McGraw Hill. ISBN 0-07-042807-7. OCLC 36417892.

21. IBM Cloud Education. (2020, August 17). *What are neural networks?* IBM. Retrieved November 7, 2022, from https://www.ibm.com/cloud/learn/neural-networks

22. Wikimedia Foundation. (2022, June 14). *Music Information Retrieval*. Wikipedia. Retrieved November 7, 2022, from https://en.wikipedia.org/wiki/Music_information_retrieval

23. Dhariwal, P., Jun, H., Payne, C., Kim, J. W., Radford, A., & Sutskever, I. (2020, April 30). *Jukebox: A generative model for music*. arXiv.org. Retrieved November 7, 2022, from https://arxiv.org/abs/2005.00341

24. Vilone, Giulia; Longo, Luca (2021). "Notions of explainability and evaluation approaches for explainable artificial intelligence". Information Fusion.

December 2021 - Volume 76: 89–106. doi:10.1016/j.inffus.2021.05.009

25. Biecek, P. (2018, July 5). *DALEX: Explainers for complex predictive models*. arXiv.org. Retrieved November 7, 2022, from https://arxiv.org/abs/1806.08915

26. Bernal, J.; Mazo, C. Transparency of Artificial Intelligence in Healthcare: Insights from Professionals in Computing and Healthcare Worldwide. Appl. Sci. 2022, 12, 10228. https://doi.org/10.3390/app122010228

27. Gunning, D.; Stefik, M.; Choi, J.; Miller, T.; Stumpf, S.; Yang, G.-Z. (2019-12-18). "XAI-Explainable artificial intelligence". Science Robotics. 4 (37): eaay7120. doi:10.1126/scirobotics.aay7120. ISSN 2470-9476. PMID 33137719.

28. Bostrom, N., & Yudkowsky, E. (2014). The ethics of artificial intelligence. The Cambridge Handbook of Artificial Intelligence, 316-334.

29. McDermid, J. A., Jia, Y., Porter, Z., & Habli, I. (2021, August 16). *Artificial Intelligence explainability: The Technical and Ethical Dimensions*. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences. Retrieved November 8, 2022, from https://royalsocietypublishing.org/doi/full/10.1098/rsta.2020.0363

30. Stupp, C. (2019, August 30). *Fraudsters used AI to mimic CEO's voice in unusual cybercrime case*. The Wall Street Journal. Retrieved November 8, 2022, from https://www.wsj.com/articles/fraudsters-use-ai-to-mimic-ceos-voice-in-unusual-cybercrime-case-11567157402

31. Villasenor, J. (2022, March 9). How to deal with ai-enabled disinformation. Brookings. Retrieved November 8, 2022, from https://www.brookings.edu/research/how-to-deal-with-ai-enabled-disinformation/

32. Malhi, A., Knapic, S., & Främling , K. (1970, January 1). *Explainable agents for less bias in human-agent decision making*. SpringerLink. Retrieved November 8, 2022, from https://link.springer.com/chapter/10.1007/978-3-030-51924-7_8

33. E. Sengupta, D. Garg, T. Choudhury and A. Aggarwal, "Techniques to Elimenate Human Bias in Machine Learning," 2018 International Conference on System Modeling & Advancement in Research Trends (SMART), 2018, pp. 226-230, doi: 10.1109/SYSMART.2018.8746946.

34. Hunkenschroer, A. L., & Luetge, C. (2022, February 8). *Ethics of ai-enabled recruiting and selection: A review and research agenda - journal of business ethics*. SpringerLink. Retrieved November 8, 2022, from https://link.springer.com/article/10.1007/s10551-022-05049-6

35. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J.,

Winter, C., … Amodei, D. (2020, July 22). *Language models are few-shot learners*. arXiv.org. Retrieved November 8, 2022, from https://arxiv.org/abs/2005.14165

36. International Music Information Retrieval Systems Evaluation Laboratory (IMIRSEL) at School of Information Sciences, University of Illinois at Urbana-Champaign. (n.d.). *Music Information Retrieval Evaluation eXchange*. 2005:Audio Genre - MIREX Wiki. Retrieved November 9, 2022, from https://www.music-ir.org/mirex/wiki/2005:Audio_Genre

37. S. Khalid, T. Khalil and S. Nasreen, "A survey of feature selection and feature extraction techniques in machine learning," 2014 Science and Information Conference, 2014, pp. 372-378, doi: 10.1109/SAI.2014.6918213.

38. Poulsen, T. (1981, June 1). *Loudness of tone pulses in a free field*. Scitation. Retrieved November 9, 2022, from https://asa.scitation.org/doi/10.1121/1.385915

39. Projects, C. to W. (2020, August 15). Wikimedia Commons. Retrieved November 9, 2022, from https://commons.wikimedia.org/wiki/Wikimedia_Commons

40. Palanisamy, K., Singhania, D., & Yao, A. (2020). Rethinking CNN Models for Audio Classification. *arXiv*. https://doi.org/10.48550/arXiv.2007.11154

41. K. J. Piczak. ESC: Dataset for Environmental Sound Classification. Proceedings of the 23rd Annual ACM Conference on Multimedia, Brisbane, Australia, 2015.

42. J. Salamon, C. Jacoby and J. P. Bello, "A Dataset and Taxonomy for Urban Sound Research", 22nd ACM International Conference on Multimedia, Orlando USA, Nov. 2014.

43. Taemin. (2021, October 29). *What is saliency map?* GeeksforGeeks. Retrieved November 9, 2022, from https://www.geeksforgeeks.org/what-is-saliency-map/

44. Simonyan, K., & Zisserman, A. (2015, April 10). *Very deep convolutional networks for large-scale image recognition*. arXiv.org. Retrieved November 9, 2022, from https://arxiv.org/abs/1409.1556

45. Y. Bengio, P. Simard and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," in IEEE Transactions on Neural Networks, vol. 5, no. 2, pp. 157-166, March 1994, doi: 10.1109/72.279181.