LOCAL INTERPRETABLE MODEL-AGNOSTIC EXPLANATIONS FOR MUSIC CONTENT ANALYSIS

Saumitra Mishra, Bob L. Sturm, Simon Dixon
Centre for Digital Music, Queen Mary University of London, United Kingdom
{ saumitra.mishra, b.sturm, s.e.dixon}@qmul.ac.uk

ABSTRACT

The interpretability of a machine learning model is essential for gaining insight into model behaviour. While some machine learning models (e.g., decision trees) are transparent, the majority of models used today are still black-boxes. Recent work in machine learning aims to analyse these models by explaining the basis of their decisions. In this work, we extend one such technique, called local interpretable model-agnostic explanations, to music content analysis. We propose three versions of explanations: one version is based on temporal segmentation, and the other two are based on frequency and time-frequency segmentation. These explanations provide meaningful ways to understand the factors that influence the classification of specific input data. We apply our proposed methods to three singing voice detection systems: the first two are designed using decision tree and random forest classifiers, respectively; the third system is based on convolutional neural network. The explanations we generate provide insights into the model behaviour. We use these insights to demonstrate that despite achieving 71.4% classification accuracy, the decision tree model fails to generalise. We also demonstrate that the model-agnostic explanations for the neural network model agree in many cases with the model-dependent saliency maps. The experimental code and results are available online. [1]

1. INTRODUCTION

Music content analysis (MCA) research aims to build systems with the sensitivity and intelligence required to work with information in acoustic environments. Recent advances in this domain have been made by leveraging large amounts of data with statistical machine learning, e.g., [5, 6]. The complexity of the resulting systems, however, makes it extremely difficult to understand their behaviours, or to predict their success in the real world.

Recent work seeks to ascribe certain functions or sensitivities to architectural elements of a trained system. For instance, analyses of deep computer vision systems find the first layer to be sensitive to edges, points and colour gradients, and deeper layers appear sensitive to higher-level concepts like faces, trees and cars [23,25,26]. Similar work for deep MCA systems has found that the first layer is sensitive to frequency bands, and deeper layers appear sensitive to timbres and temporal patterns, e.g., [3, 6]. In a different direction, other research focuses on approaches to explain individual predictions. One approach to explain individual predictions substitutes complex black-box models with inherently interpretable models whose predictions can be summarised by simple if-else rules [11,24]. Other methods use sensitivity analysis [7] or Taylor series expansion [15] to analyse the prediction function locally. Sensitivity analysis aims to capture the local behaviour of the prediction function when the input dimensions are perturbed. Variants of this approach include saliency maps [20], explanation vectors [1], "horse" detection [22] and local interpretable model-agnostic explanations (LIME) [17]. In this paper, we focus on extending LIME for MCA.

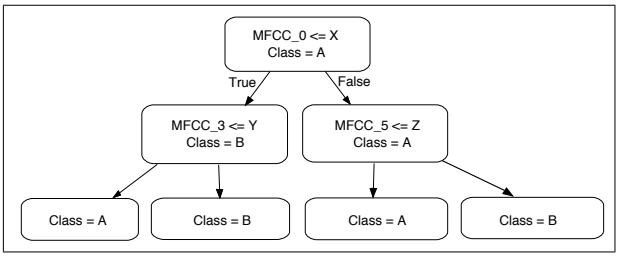

Figure 1: A binary decision tree for classifying audio using the values of three MFCC feature dimensions.

LIME is an algorithm that provides instance-based explanations to predictions of any classifier. These explanations are locally faithful to the instance, independent of the classifier model type, and are learned over interpretable representations of the instance. For example, for an e-mail classification system, LIME generates a list of words of an e-mail as an explanation for its classification to some category. To produce the explanation, LIME approximates the classifier locally with an interpretable model (e.g., sparse linear models, decision trees).

We introduce three different versions of explanations to apply LIME to MCA. We call this extended framework as Sound LIME (SLIME). Each version works in the time, frequency and time-frequency domains, respectively. SLIME pinpoints the time or time-frequency region that contributes most to a decision. This transforms a non-intuitive feature-based classifier decision into a more intuitive temporal and spectral description. We demon-

[1] https://code.soundsoftware.ac.uk/projects/SoundLIME

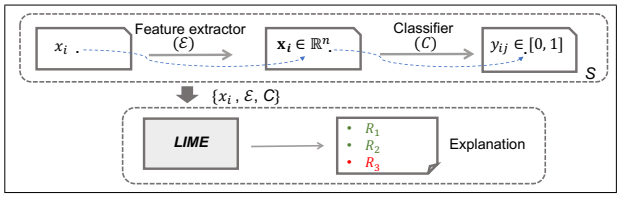

Figure 2: Schematic representation of LIME explaining why an MCA system S applies label j to instance x_i with probability y_{ij}.

strate SLIME for three trained singing voice detection systems, and show how the generated explanations are useful in gaining insight into model behaviour, and in identifying an untrustworthy model that fails to generalise.

2. MOTIVATION

Consider a simple MCA system, the classification component of which is the binary decision tree (BDT) shown in Fig. 1. The input to this system is a T-sec excerpt of audio, from which the system extracts D Mel-frequency cepstral coefficients (MFCC) [4]. This D-dimensional feature vector is labeled by the system as either "class A" or "class B" based on the values in specific dimensions. The particular dimensions, and the thresholds of the decisions, are found through training.

A binary decision tree is a transparent classifier because one can trace the reason for a particular outcome - in this case in terms of the MFCC coefficients and thresholds. As shown in Fig. 1, if the value of the zeroth MFCC is less than X and that of the third MFCC is less than Y, then this system classifies the instance as "class A". What does this mean in terms of the qualities of the input sound, however? Why X? Is this a real-world general principle? Or does it arise from a peculiarity of the training dataset?

MFCCs were introduced for speech recognition [4], but have been argued as suitable for machine music listening [9, 12]. Extracting MFCC features from audio involves windowing (typically on the order of 10-100 ms), a Mel-scale based smoothing of the log magnitude spectrum, and discrete cosine transform (DCT)-based compression. Although MFCC features are pseudo-invertible [2], they are difficult to interpret in terms of the qualities of the underlying sound. This comes in part from frequency bin grouping and the log magnitude operations, which destroy the bijective mapping between the audio and its spectrum.

One might still roughly approximate the meaning of particular MFCCs: low MFCC dimensions relate to broad spectral structures (e.g., formants); high MFCC dimensions relate to fine spectral structures (e.g., pitch and harmonics); and the zeroth MFCC relates to the energy of a signal. But, as shown in Fig. 1, values along several MFCC dimensions and their thresholds jointly contribute to a prediction. This combination makes interpretation even harder. It is hard to understand what audible qualities are captured by the combination of the zeroth MFCC with either the third or the fifth MFCC. Thus, though the decision tree has clear decision rules, they are not easy to relate to audible qualities of inputs. With other machine learning systems, e.g., deep neural networks or support vector machines, this task becomes harder still. This motivates the use of "interpretable representations" for explaining system behaviours for specific inputs.

3. INTERPRETABLE EXPLANATIONS FOR MUSIC CONTENT ANALYSIS

We first present the local interpretable model-agnostic explanations (LIME) proposed in [17]. We then extend it to working with MCA systems.

3.1 Summary of LIME [17]

Section 2 shows how the rules guiding a classifier's output can be difficult to interpret in terms of content, even for transparent classifiers. This interpretability becomes increasingly difficult when the model becomes complex (e.g., support vector machine) or the feature extraction is replaced by feature learning (e.g., convolutional neural network). LIME uses an interpretable representation of data to maintain interpretability in the generated explanations. Such explanations are easier because they show a more direct mapping between the input and its prediction.

LIME is an algorithm that generates interpretable, locally faithful and model-agnostic explanations to predictions of any classifier. Fig. 2 depicts a high-level overview of what LIME aims to perform. LIME helps illuminate reasons for a system S applying label j to instance x_i with probability y_{ij}. For example, for the input x_i, LIME lists three reasons: R_1, R_2 and R_3, to explain the prediction. R_1 and R_2 are positively correlated with the decision and R_3 is negatively correlated.

Locally faithful explanations refer to capturing the classifier behaviour in the neighbourhood of the instance to be explained. To learn a local explanation, LIME approximates the classifier's decision boundary around a specific instance using an interpretable model. LIME is *model-agnostic*, i.e., it considers the model as a black-box and makes no assumptions about the model behaviour. This makes LIME applicable to any classifier.

Formally, let $C : \mathbb{R}^n \rightarrow \mathbb{R}$ be a classifier, mapping a feature vector to a class label. For a feature vector $\mathbf{x}_i = \varepsilon(x_i)$, denote $y_{ij} = C(\mathbf{x}_i)$ as the probability that $\mathbf{x}_i$ takes the class label j. Define a sequence $\mathcal{X}_i$, which is composed of elements that are in some sense meaningful with respect to the classification of the instance x_i. For example, for a text classification system, $\mathcal{X}_i$ could be the sequence of unique words in e-mail. LIME defines an *interpretable space* $\mathcal{T} = \{0, 1\}^{|\mathcal{X}_i|}$, where its kth dimension corresponds to the kth element of $\mathcal{X}_i$. Then $\mathbf{x}'_i \in \mathcal{T}$ is the *interpretable representation* of x_i. Thus, LIME transforms the input instance x_i to a binary vector $\mathbf{x}'_i$ whose elements correspond to presence and absence of elements of $\mathcal{X}_i$.

LIME defines an *interpretable explanation* as a model $g \in G$, where G denotes a class of interpretable models (e.g., linear models, decision trees). LIME learns a model g over the interpretable space by the optimisation:

$$\min_{g \in G} \; L(C, g, \rho_{x_i}) + \Delta(g) \qquad (1)$$

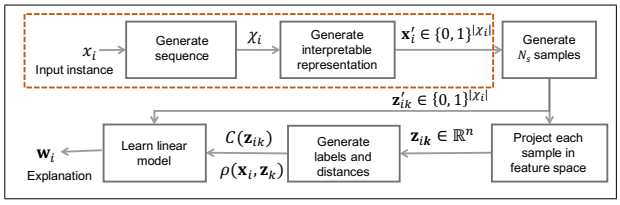

Figure 3: Functional block diagram of SLIME depicting the steps in generation of the explanation $\mathbf{w}_i$ for the prediction of the instance x_i.

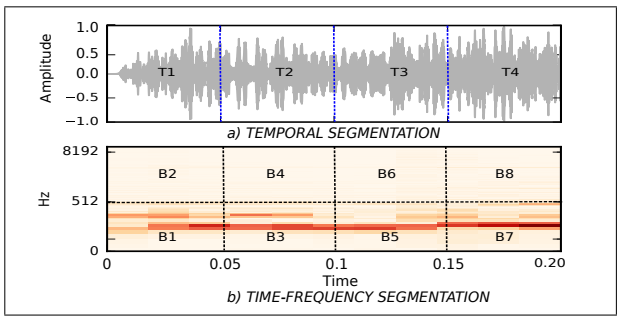

Figure 4: Segmentation based sequence generation for SLIME. (a) Temporal segmentation of instance x_i into four super samples (T_i), each of duration 50 ms. (b) Time-frequency segmentation of instance x_i into 8 blocks (B_i).

where $L(C, g, \rho_{x_i})$ is a locally-weighted loss function that for an instance x_i measures how well the model g approximates the classifier C in the neighbourhood defined by ρ_{x_i}, and $\Delta(g)$ is a measure of model complexity (e.g. sparsity in linear models). Thus, LIME minimises this function to explain why C maps x_i to class label j.

3.2 Extending LIME to MCA

Fig. 3 depicts the functional block diagram of SLIME. This consists of two components: the first one is our contribution (dotted box in Fig. 3), which defines interpretable sequences for an input audio. The second one is the LIME algorithm that uses the defined representations to generate explanations.

The first step in SLIME is to define a sequence denoted $\mathcal{X}_i$ from an input instance x_i. We define three kinds of sequences: temporal $\mathcal{X}_i^t$, spectral $\mathcal{X}_i^f$ and time-frequency $\mathcal{X}_i^{tf}$. We call each element of $\mathcal{X}_i^t$ a *super sample*, which we generate by temporal partitioning of x_i. For example, the instance shown in Fig. 4(a) is uniformly segmented into four super samples each notated T_i. Hence, $\mathcal{X}_i^t = (T_1, T_2, T_3, T_4)$. Similarly, each element of $\mathcal{X}_i^f$, notated A_i is a spectral magnitude in a corresponding frequency bin, obtained by the Fourier transform of x_i. Hence, $\mathcal{X}_i^f = (A_1, A_2, A_3, ...)$. Lastly, each element of $\mathcal{X}_i^{tf}$, notated B_i is obtained by segmenting the magnitude spectrogram of the input instance, both along the time and frequency axes. For example, in Fig. 4(b) the spectrogram of the instance is non-uniformly segmented into eight time-frequency blocks. Hence, $\mathcal{X}_i^{tf} = (B_1, B_2, ..., B_7, B_8)$. We call each element of a sequence as an *interpretable component*. Thus, for a temporal sequence each interpretable component is a supersample and for spectral and time-frequency sequences each interpretable component is a spectral bin and time-frequency block, respectively.

The next step is to map the input instance with feature representation denoted as $\mathbf{x}_i \in \mathbb{R}^n$ to its interpretable representation denoted as $\mathbf{x}'_i \in \{0,1\}^{|\mathcal{X}_i|}$. Thus, each of the above mentioned sequences is used to define an interpretable space $\mathcal{T}$ and an interpretable representation $\mathbf{x}'_i$. This creates three interpretable representations for the input instance x_i. We denote temporal, spectral and time-frequency interpretable representations as $\mathbf{x}_i^{t\prime}$, $\mathbf{x}_i^{f\prime}$ and $\mathbf{x}_i^{tf\prime}$ respectively. These representations provide us three ways of understanding a prediction, each highlighting the temporal, spectral or time-frequency segments of the instance influencing the prediction most.

To find an explanation, SLIME approximates the classifier $C : \mathbb{R}^n \rightarrow \mathbb{R}$ with a linear model $\{g(\mathbf{z}') = \mathbf{w}^T\mathbf{z}'; \mathbf{z}' \in \mathcal{T}\}$. To do this SLIME first generates N_s samples from $\mathcal{T}$ in a way that depends on $\mathbf{x}'_i$, i.e., randomly setting to zero the dimensions of $\mathbf{x}'_i$. Hence, for the interpretable sequence $\mathcal{X}_i^t$ in Fig. 4(a), one possible $\mathbf{z}_i^{t\prime}$ = (1, 0, 1, 0). This synthetic sample indicates the absence of super samples T_2 and T_4. Formally, for an instance with N_s super samples, a total of 2^{N_s} synthetic samples exists. With an assumption that there exists a surjective map from $\mathbb{R}^n$ to $\mathcal{T}$, each synthetic sample is projected to $\mathbb{R}^n$, weighted using an exponential kernel learned over cosine distance (we used the same ρ_{x_i} as in [17]) and mapped to its corresponding probability $C(z)$. SLIME learns the linear model g_t by minimising the squared loss and model complexity as in (1) over this dataset of synthetic samples and their probabilities. Formally, denote the kth sample as $\mathbf{z}'_k$ and its projection $\mathbf{z}_k$. Define a weight function $\rho_{x_i} : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$. The locally-weighted loss used by SLIME is given by

$$L(C, g, \rho_{x_i}) = \sum_{(\mathbf{z}'_k, \mathbf{z}_k) \in Z} \rho(\mathbf{x}_i, \mathbf{z}_k)[C(\mathbf{z}_k) - g(\mathbf{z}'_k)]^2 \quad (2)$$

Similarly, SLIME randomly samples $\mathbf{x}_i^{f\prime}$ and $\mathbf{x}_i^{tf\prime}$ to learn the linear models g_f and g_{tf}, respectively. Each of these models provides interpretable explanations in terms of their learned weights. The magnitude of the coefficients relates to the importance of the temporal segment (super sample) or the spectral component (bin frequency) or the time-frequency block in the classification of $\mathbf{x}_i$. Thus, if w_1 and w_2 denote the coefficients of super samples T_1 and T_2 respectively, then $|w_1| \geq |w_2|$ implies super sample T_1 has more influence on a classification prediction than T_2. Similarly, the polarity of regression weights refers to the correlation between the segment and the classifier prediction. For example, if $w_1 < 0$ and $w_2 > 0$, then the temporal segments T_1 and T_2 are negatively and positively correlated with the classifier prediction. The weight function ρ_{x_i} controls the contribution each synthetic sample has in the learned model g. Thus, a distant sample in interpretable space $\mathcal{T}$ will have lower contribution to g facilitating better learning in cases where the random sampling produces samples with highly imbalanced class distributions.

Classifier	*Acc[%]*	*Prec.*	*Recall*	*F- score*
Decision tree	71.4	0.72	0.81	0.75
Random forest	76.3	0.75	0.88	0.79

Table 1: Singing voice class evaluation results for the two selected shallow SVD systems (a) Binary decision tree of depth 8 and information gain as the split criterion (b) Random forest of 64 trees, each with depth 16.

4. DEMONSTRATION

We now use SLIME to explain the predictions of three singing voice detection (SVD) systems that classify an audio excerpt into two categories: music without singing voice, and music with singing voice. Two systems are based on a shallow architecture proposed in [10]. The other one is based on hierarchical feature learning [19].

4.1 Explaining Predictions of Shallow Vocal Detectors

Several shallow vocal detection systems have been proposed [10, 13, 16, 18]. We adapt the method proposed in [10] that uses only MFCC features to reach state of the art performance. Our system calculates FFTs on a frame size of 200 ms with 50% overlap at a sampling frequency of 22050 Hz. It uses a set of 30 Mel-filters to extract 30 MFCC coefficients (including the 0^{th}) and their first-order derivatives from each audio frame using *Librosa* [14]. The system performs classification over a 1 sec excerpt, hence it calculates the median and standard deviation of the 60 dimensional vector over five frames [18], constructing a feature vector of 120 dimensions.

We train two systems: the first (S_1) combines a binary decision tree (BDT) with the feature vector from above and the second (S_2) replaces the BDT with a random forest (RF) classifier. The *Jamendo* dataset, introduced in [16] is used to train, validate and evaluate both the models on three non-overlapping sets. Table 1 reports the results of the evaluation for singing voice class. The vocal detection systems designed using the BDT and RF classifiers achieve an overall accuracy of 71.4% and 76.3%, respectively. The vocal class occupies 57.5% of the test dataset which suggests that these two systems may have learnt some representation of singing voice that helps to detect vocals. We now apply SLIME to determine if these systems are trustworthy [22]. In other words, are the vocal predictions caused by content where there actually is voice?

In order to generate temporal explanations, we segment the instance (1 sec) into ten super samples, each of 100 ms duration. We first generate 1000 samples in the interpretable space. We then approximate each classifier's decision boundary in a neighbourhood of the instance by a linear model learnt over the interpretable space. The number of interpretable components needed to explain an instance may vary from one instance to the other, but to reduce the model complexity ($\Delta(g)$ in (1)), we generate explanations with a fixed number of components. To do this we first use the synthetic dataset of perturbed samples and their probabilities to select the top-3 super samples by forward selection, and then learn a linear model [17].

Id.	*Dur. (s)*	*Prob-Vocal*		*SS-Pred.*		*SS-True*
		BDT	*RF*	*BDT*	*RF*	
41	1.0	0.97	0.85	6,7,9	2,0,7	0-9
178	1.0	0.86	0.86	9,8,4	9,6,0	0-9
58	0.4	0.80	0.76	6,5,3	0,2,6	0-3
124	0.4	0.92	0.84	0,4,6	6,9,8	6-9

Table 2: Instance-based temporal explanations generated by SLIME. Id: instance index, Dur: vocal duration, SS: super samples, Prob-Vocal: probability assigned by the SVD system that the instance contains singing voice, SS-Pred: super sample indices that are the most influential upon the classification of the input instance to vocal class, SS-True: super sample indices that actually contain singing voice.

Table 2 reports the temporal explanations generated by SLIME for four instances extracted from the "03 - Say me Good Bye.mp3" test file in the Jamendo dataset. The super samples are arranged in the decreasing order of influence on the prediction. The magnitude of the weights learned for each super sample determines the influence it has on the prediction. This analysis of the temporal explanations helps to gain insight about how the models are forming their predictions. For example, instance 41 is correctly predicted by both the models (true positive). But, the temporal explanations for both the models are very different. The same is the case with another instance 178. Listening to all the predicted super samples for instances 41 and 178, highlights an interesting observation. For most of the predicted super samples for the decision tree model there is a presence of 'strong' instrumental onset along with the singing voice. Thus, it might be the case that instead of "listening" to the singing voice in the super sample, the decision tree model is paying attention to instrumental onset.

To verify the above hypothesis, we select true positive instances that have instrumental music and singing voice as separate temporal sections. We apply SLIME to two such instances: 58 and 124, which have singing voice in the first and last 400 ms, respectively. The temporal explanations generated for the BDT highlight that even though the prediction score is high for the model, the super samples it believes to contain singing voice have only instrumental music in most of the explanations. This raises questions about the generalisation capability of such a model. Based on the explanations generated for the RF model, it appears that the model is looking at the right temporal sections to form a prediction. Thus, the temporal explanations are helpful in identifying an untrustworthy model.

Temporal explanations help to understand the predictions but under some limitations. First, for the explanations to be clearly audible super samples should be at least 100ms long. Second, for the cases as in instance 41, where singing voice and instrumental music are present for complete duration, temporal explanations do highlight which temporal sections are useful for prediction but not what in that section is important. One way to solve this problem is to use SLIME to generate the spectral or time-frequency explanations as demonstrated below.

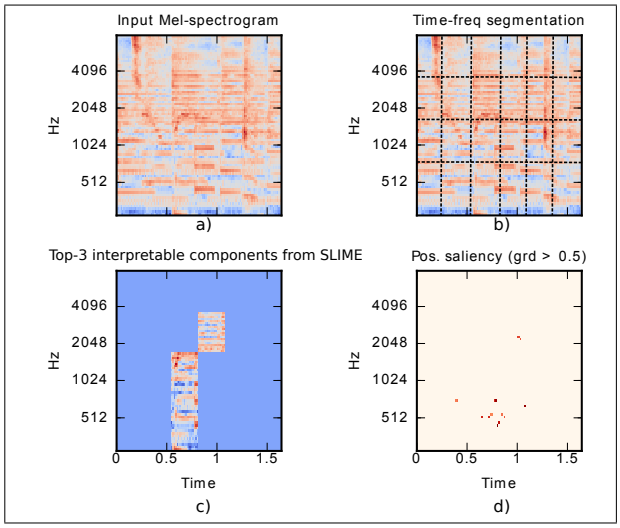

Figure 5: Comparing the positive explanation from SLIME with the positive saliency map for a 1.6s excerpt.

4.2 Explaining Predictions of a Deep Vocal Detector

We now demonstrate SLIME working with the convolutional neural network (CNN)-based system proposed in [19]. We generate time-frequency explanations for the predictions of the system and compare the generated explanations with the saliency maps [20, 21, 26]. Due to space restriction we have skipped the demonstration of spectral explanations, but such explanations can be easily derived from time-frequency explanations by expanding the temporal analysis window to full length of the excerpt.

The system proposed in [19] takes in a Mel-spectrogram representation of a 1.6 second audio excerpt and returns the probability that it contains singing voice. In order to explain the predictions of the system, we map the Mel-spectrogram to the time-frequency interpretable representation proposed in subsection 3.2. We segment the time-frequency axis of the input in 6 and 4 segments, respectively. Thus, the temporal axis of each of the first 5 segments is 266 ms in duration and that of the last segment is 280 ms. We aim to keep the temporal axis of the resulting interpretable components long enough to facilitate audition in the temporal domain. Similarly, segmentation along the frequency axis results in 4 spectral sections, each with 20 spectral bins. Thus, the input Mel-spectrogram is mapped to a sequence of time-frequency blocks $\mathcal{X}_i^{tf} = (B_1, ..., B_{24})$, where each block represents a dimension in the interpretable space. Fig. 5(a), (b) depict the Mel-spectrogram and its time-frequency segmentation, respectively for an input excerpt from "03 - Say me Good Bye.mp3" file from the Jamendo test dataset.

SLIME generates 2000 samples in the neighbourhood of the input, approximates the non-linear decision boundary by a linear model, and selects the top-3 interpretable components (time-frequency blocks) with the highest positive weights. Fig. 5(c) depicts the positive explanation for the prediction of the audio excerpt. We call an explanation positive if the weights of the interpretable components in the explanation are positive. The input excerpt chosen for analysis has singing voice with musical accompaniment for the first 900 ms and only musical accompaniment for the last 700 ms. We invert the time-frequency blocks in the explanation to temporal domain and on listening find that all the components in the explanation have the presence of singing voice. This raises confidence in the predictions of the model. Moreover, all the components in the negative explanation (not shown due to space restriction), fall in the temporal sections after 1s. This indicates that the time-frequency segments containing only instrumental music are negatively correlated with the classifier prediction. This also seems to be correct behaviour. Thus, the time-frequency explanations help to understand what sections in the input are influencing the prediction most.

We now compare SLIME-based explanations with saliency maps. Saliency maps, like time-frequency explanations, are tools to analyse black-box neural network models. They highlight how each input dimension influences the prediction. The gradient of the output prediction with respect to each input dimension is calculated to compute the saliency maps [20]. Thus, they depict the effect of modifying the input along any dimension, on the network prediction. Instead of allowing all the gradients to flow back, techniques proposed in [21, 26] only allow the positive gradient to flow back resulting in cleaner visualisations. Using the technique proposed in [26], we employ a leaky-ReLU non-linearity [8] in the backward path to reduce the magnitude of the negative gradients flowing back. We compare the positive time-frequency explanations with the positive saliency map. This map will highlight the input dimensions that are positively correlated with the classifier prediction. Not all the dimensions influence the predictions equally, thus we select only those dimensions whose normalised gradient is more than 0.5. We generate such maps for the output layer of the network. Fig. 5(d) shows the thresholded positive saliency map.

It is important to note that saliency maps highlight individual dimensions in the input while SLIME based explanations are time-frequency blocks. One way to compare the two is by visually verifying whether all the dimensions highlighted by the saliency maps are captured in the explanations created by SLIME. A visual comparison for the example in Fig. 5 shows that SLIME's explanation includes most of the key dimensions highlighted by the saliency map. Numerically we measure how many dimensions highlighted by the saliency map are enclosed in the explanation generated by SLIME. For the audio excerpt shown in Fig. 5, this agreement is 62.5%. We expand this analysis to a set of 1349 randomly chosen excerpts from the Jamendo test dataset. We found that on an average SLIME achieves 46.50 % numerical agreement when compared with the positive saliency maps. Instance-based analysis reveals that in some instances the numerical agreement is 100%, but there are cases where this number is less than 10%. One possible explanation for this is the shape of decision boundary near the instance. If the decision boundary is highly non-linear, approximating it with a linear model will result in poor explanations from SLIME.

We have not performed an exhaustive comparison (by

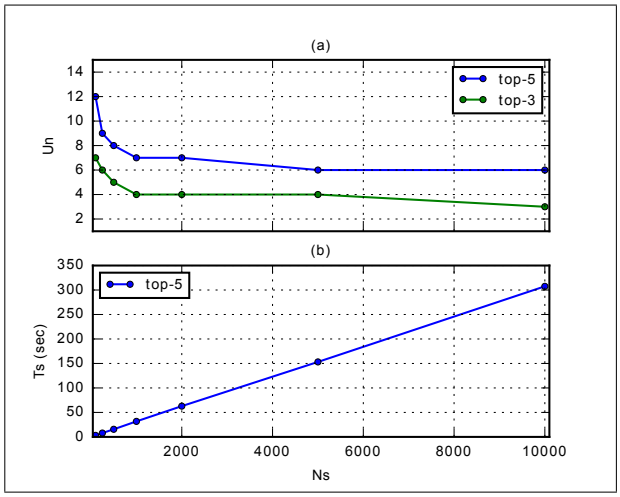

Figure 6: Plotting the effect of the number of samples (N_s) on (a) the stability of generated explanations (b) the time taken to generate them.

varying the preset factors, e.g. threshold, number of components) between the two techniques. The above analysis aims to provide an estimate about the performance of model-agnostic SLIME against a model-dependent technique for some preset values. It is obvious that the numerical agreement will be high if the constraints are softer and vice versa. Although saliency maps are accurate in highlighting the input dimensions that are influential to a classifier's output, they can suffer from lack of temporal context around the dimensions. On the other hand, SLIME-based explanations can be readily inverted to an acoustic form for audition, which may provide additional insights into how a classifier is forming its prediction for an input.

4.3 Discussion on the Number of Samples (N_s)

As discussed in subsection 3.2, to explain a prediction LIME generates N_s samples in the interpretable space ($\mathcal{T}$). In [17] there is no discussion about how many samples should be used to generate each explanation. We believe that exploring this is important for two reasons. First, it affects the time taken (T_s) to generate an explanation. Second, it affects the stability of the generated explanation. Ideally, an explanation should remain the same (at least the interpretable components, but their order and weights might change) even on multiple iterations of applying LIME to the same instance. But, empirically we find that the generated explanations do change on multiple iterations. This happens because LIME samples randomly in $\mathcal{T}$. In this section we seek to understand the effect of N_s on the stability of the explanations and on the time to generate one explanation.

For the experiment, we use the trained model, dataset and SLIME set-up from subsection 4.2. We randomly select 5 excerpts from each test file in the Jamendo dataset. We apply SLIME to generate explanation for the prediction of each excerpt in a batch of 80 and select the top-k interpretable components per explanation (we try k = 3 and 5). We iterate this process 5 times, each time randomly sampling 80 excerpts, generating explanations and selecting the top-k interpretable components.

We define the *stability* of an explanation to be inversely proportional to the number of unique interpretable components (U_n) from the sequence $\mathcal{X}_i$ that appear in explanations generated with m iterations. For example, if we apply SLIME $m = 2$ times to an instance and select the top-3 interpretable components in each iteration. Say the selected time-frequency segments are denoted as sets $\xi_1 = \{B_1, B_2, B_3\}$ and $\xi_2 = \{B_2, B_6, B_5\}$. Then U_n = 5, as B_2 appears twice in 6 components. To understand the effect of N_s on the stability of explanations, we generate 5 explanations for each of the 80 excerpts in the randomly sampled batches. We calculate the value of U_n in all the 5 explanations for each excerpt and plot the average result over 5 batches for a given value of N_s. Fig. 6(a) reports the results of the experiment. The result shows that U_n is inversely related to N_s, and thus the stability of the generated explanations is proportional to N_s. The result also shows that exhaustive search of the interpretable space $\mathcal{T}$ is not needed to generate stable explanations.

We also record the average time taken to generate one explanation for a given value of N_s. Results are generated by running SLIME on a computer with 1.6 GHz Intel core i5 processor and 8 GB memory and are reported in Fig. 6(b). Results show that T_s increases linearly with N_s, reaching to a maximum of around 5 mins for an explanation generated with N_s = 10k. The reported time includes the time taken for prediction by the CNN. These results suggest that selecting a suitable N_s depends on the trade-off between the stability of an explanation and the time-taken to generate it. In our experiment N_s = 1000 seems to be a good trade-off.

5. CONCLUSION

In this work we proposed SLIME, an algorithm that extends the applicability of LIME [17] to MCA systems. We proposed three versions of SLIME and demonstrated them with three types of singing voice detection systems to generate temporal and time-frequency explanations for the predictions of specific instances. We see that the temporal explanations generated by SLIME are helpful for revealing how the BDT is making decisions based on content that does not contain singing voice despite possessing high classification accuracy for the selected instances. Such issues cast doubt on the generalisability of the model. We also demonstrated that the analysis of time-frequency explanations is helpful to gain trust in the CNN based SVD system. We compared SLIME based explanations with saliency maps for the neural network model and the results suggest that model-agnostic SLIME based explanations agree in many cases with saliency maps.

In future we would like to apply SLIME to other MCA systems. We also plan to experiment with improved interpretable representations that will be created around audio "objects". We believe that the improved representations will assist in better understanding of the behaviour of the underlying machine learning model.

6. ACKNOWLEDGEMENTS

This work is supported in part by AHRC Grant Agreement no. AH/N504531/1. We would like to thank Jan Schlüter for sharing his implementation of the neural network based SVD system. We also thank the anonymous reviewers for their valuable comments and suggestions.

7. REFERENCES

[1] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, and K.-R. Muller. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11:1803–1831, 2010.

[2] L. E. Boucheron and P. D. Leon. On the inversion of Mel-frequency cepstral coefficients for speech enhancement applications. In *Proc. of IEEE Intl. Conf. on Signals and Electronic Systems (ICSES)*, 2008.

[3] K. Choi, G. Fazekas, and M. B. Sandler. Explaining deep convolutional neural networks on music classification. *arXiv preprint arXiv:1607.02444*, 2016.

[4] S. B. Davis and P. Mermelstein. Comparison of parametric representation for monosyllabic word recognition in continously spoken sentences. *IEEE Trans. on Acoustics, Speech and Signal Processing.*, 28(4):357–366, 1980.

[5] L. Deng and D. Yu. *Deep Learning: Methods and Applications*. Now Publisher, 2014.

[6] S. Dieleman and B. Schrauwen. End-to-end learning for music audio. In *Proc. ICASSP*, 2014.

[7] M. Gevrey, I. Dimopoulos, and S. Lek. Review and comparison of methods to study the contribution of variables in artificial neural network models. *Ecological Modelling*, 160(3):249–264, February 2003.

[8] A. Jannun, A. Maas, and A. Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[9] J. H. Jensen, M. G. Christensen, D. P. W. Ellis, and S. H. Jensen. Quantitative analysis of a common audio similarity measure. *IEEE Trans. on Audio, Speech, and Language Processing*, 17(4):693–703, 2009.

[10] B. Lehner, R. Sonnleitner, and G. Widmer. Towards light-weight, real-time-capable singing voice detection. In *Proc. ISMIR*, 2013.

[11] B. Letham, C. Rudin, T. H. McCormick, and D. Madigan. Interpretable classifiers using rules and Bayesian analysis: Building a better stroke prediction model. *Annals of Applied Statistics*, 9(3):1350–1371, 2015.

[12] B. Logan. Mel frequency cepstral coefficients for music modeling. In *Proc. ISMIR*, 2000.

[13] M. Mauch, H. Fujihara, K. Yoshii, and M. Goto. Timbre and melody features for the recognition of vocal activity and instrumental solos in polyphonic music. In *Proc. ISMIR*, 2011.

[14] B. McFee, C. Raffel, D. Liang, D. P. W. Ellis, M. McVicar, E. Battenberg, and O. Nieto. librosa: Audio and music signal analysis in python. In *Proc. of the 14th Python in Science Conference (SCIPY)*, 2015.

[15] G. Montavon, S. Bach, A. Binder, W. Samek, and K. R. Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

[16] M. Ramona, G. Richard, and B. David. Vocal detection in music using support vector machines. In *Proc. ICASSP*, pages 1885–1888, 2008.

[17] M. T. Ribeiro, S. Singh, and C. Guestrin. Why should I trust you?": Explaining the predictions of any classifier. In *Proc. of the ACM Conf. on Knowledge Discovery and Data Mining (KDD)*, 2016.

[18] M. Rocamora and O. Herrera. Comparing audio descriptors for singing voice detection in music audio files. In *Brazilian Symposium on Computer Music*, volume 26, page 27, 2007.

[19] J. Schlüter and T. Grill. Exploring data augmentation for improved singing voice detection with neural networks. In *Proc. ISMIR*, 2015.

[20] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *Proc. ICLR Workshop*, 2014.

[21] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. In *Proc. of the Intl. Conf. on Learning Representations (ICLR) Workshop*, 2015.

[22] B. L. Sturm. A simple method to determine if a music information retrieval system is a "horse". *IEEE Trans. on Multimedia*, 16(6):1636–1644, 2014.

[23] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *Proc. ICLR*, 2014.

[24] F. Wang and C. Rudin. Falling rule lists. In *Proc. AISTATS*, 2015.

[25] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding Neural Networks Through Deep Visualization. In *Proc. of the Intl. Conf. on Machine Learning (ICML) Deep Learning Workshop*, 2015.

[26] M. D. Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks. In *Proc. of the European Conf. on Computer Vision*, 2014.