"What are You Listening to?" Explaining Predictions of Deep Machine Listening Systems

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

***Abstract*—Researchers have proposed methods to explain neural network predictions by building explanations either in terms of input components (e.g., pixels in an image) or in terms of input regions (e.g., the area containing the face of a Labrador). Such methods aim to determine the trustworthiness of a model, as well as to guide its improvement. In this paper, we argue that explanations in terms of input regions are useful for analysing machine listening systems. We introduce a novel method based on feature inversion to identify a region in an input time-frequency representation that is most influential to a prediction. We demonstrate it for a state-of-the-art singing voice detection model. We evaluate the quality of the generated explanations on two public benchmark datasets. The results demonstrate that the presented method often identifies a region of an input instance that has a decisive effect on the classification.**

***Index Terms*—Deep neural networks, visualisation, interpretable machine learning, machine listening**

I. Introduction

Deep neural networks (DNNs) have recently demonstrated remarkable success in several pattern recognition use cases [1]. Although highly accurate, these models are "black-boxes", as theoretically and empirically we know little about their inner functioning. As DNNs are susceptible to adversarial perturbations [2], understanding their behaviour is crucial to gain trust in their predictions. Moreover, such an analysis may also assist in refining model architectures [3].

Researchers have proposed several post-hoc model visualisation methods to understand how a DNN model forms its decisions [4]. One category of methods focuses on analysing the global behaviour of the model. For example, *Activation-Maximisation*, a widely used method for global analysis, involves generating patterns in the input space (e.g., image) to maximally activate a specific neuron or a layer in the network [5]. Such an analysis is useful, but often for deeper layers, the generated patterns are difficult to interpret due to the multi-faceted nature of neurons [6].

In another direction, there are methods that limit the analysis to individual examples (local analysis) and focus on understanding the input dimensions that contribute to a prediction. The vast majority of such methods use variants of *sensitivity analysis* to capture the effect of modifying a dimension or a group of dimensions on the final prediction. For example, the methods using gradient-based sensitivity analysis [7] produce attribution (or saliency) maps that highlight the relevance of each dimension towards a prediction. On the other hand, explanations generated using occlusion [3] or model approximation [8] identify the input regions in favour of (or against) a prediction.

In this work, we consider the problem of local analysis for deep machine listening systems that classify input audio excerpts (or frames) into pre-defined categories. Previous efforts in this direction explain a prediction by assigning a relevance score either to the individual bins [9] or to the input regions (a collection of bins) [10]. We argue that highlighting relevant input bins can be less interpretable than identifying influential regions due to three key reasons: (1) saliency maps are noisy [11] and often need special methods to generate cleaner visualisations [12], which may not always be possible; (2) some saliency methods generate inconsistent explanations [13]; and (3) the lack of context around individual bins makes the attribution maps non-audible.

We introduce a novel method to automatically discover a region (a group of bins) in an input excerpt that contributes maximally towards a prediction. An existing method for local analysis with region-level granularity [10] requires information about the number of interpretable components and a segmentation methodology (e.g., uniformly segment input into N time-frequency blocks). The generated explanations may highlight non-contiguous components, which sometimes makes the interpretation of the explanation difficult. Our method does not need any auxiliary information and will generate contiguous regions as explanations.[1] Moreover, our method generates an explanation significantly more quickly than the one discussed in [10].

We evaluate the method we propose for a state-of-the-art singing voice detection (SVD) model that classifies an input mel spectrogram excerpt into predefined categories (instrumental music with and without singing voice) [14]. The empirical results suggest that the presented method assists in localising the influential input region to a fair degree of accuracy. Classification using only the identified regions is unchanged from the original classification in more than 80% of cases randomly selected from two benchmark singing voice detection datasets. The experimental code and results are available online.[2]

[1] Ribeiro et al. [8] earlier proposed homologous "super-pixel" based explanations for images.

[2] https://github.com/saum25/EUSIPCO-2018

ISBN 978-90-827970-1-5

II. Proposed Approach

Our method uses feature inversion to generate a local explanation for any prediction. Feature inversion [15], [16] involves methods to invert feature vectors, either handcrafted or extracted by a deep discriminator model D (e.g., convolutional neural network (CNN)). In this work, we focus on inverting features a deep model extracts at any layer. During discriminative learning, each hidden layer of D combines lower-level features to build higher-level representations and in the process retains only the features relevant to the discrimination task. Thus, inverting a feature vector extracted at a layer l will assist in visualising and understanding the features D preserves at that layer. For example, Dosovitskiy et al. [16] show that AlexNet [17] preserves the colour and approximate location of an object in its last fully connected layer.

We employ this idea of feature inversion to locate a region in an input time-frequency representation that contributes the most to a prediction. It is conceivable that a model predicts by using features preserved in its deepest layer (just before the Softmax layer). A model learns such features automatically, but if we invert a feature vector extracted at the deepest layer, then the inverted representation will highlight the corresponding region in the input space that a model considers important for the task. This may suggest that the discriminative features a model uses for classification are the most prominent in the highlighted region of an input. For example, in an image recognition model [17], if a model classifies an image to the 'Cat' class by looking at the face of the cat (assumption), then an inverted representation should highlight the cat's face more prominently (explanation) than other image components. Thus, in order to explain a prediction, we propose to invert the feature vector at the last hidden layer of D and convert the generated representation to a binary mask that masks the input. The unmasked section in the input will highlight the region that maximally influences a model's prediction. The method we propose consists of two steps.

- Step 1 involves training an inversion model $G : \mathbb{R}^d \rightarrow \mathbb{R}^n$. Thus, given an i^{th} audio excerpt $\mathbf{x}_i \in \mathbb{R}^n$ and a pre-learned representation function $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^d$, G maps the d-dimensional feature vector $\Phi(\mathbf{x}_i)$ (from the last hidden layer) to an inverted representation $\mathbf{x}_i^* \in \mathbb{R}^n$.
- Step 2 involves using G to explain why D classified $\mathbf{x}_i$ to a category C. This step firstly generates an inverted representation $\mathbf{x}_i^*$ using G, and later uses the explanation generator E and $\mathbf{x}_i^*$ to generate an explanation $\mathbf{x}_i^{exp}$.

A. Step 1: Training an Inversion Model

Recently, researchers have proposed two methods to invert deep feature vectors. Mahendran et al. [15] introduced a method that performs inversion by iteratively optimising a randomly sampled input $\tilde{\mathbf{x}}_i \in \mathbb{R}^n$. The objective function minimises the squared L_2 distance between a given feature vector $\Phi(\mathbf{x}_i)$ and its current version $\Phi(\tilde{\mathbf{x}}_i)$. Such an unconstrained optimisation often leads to fooling examples [18]. Thus, they also proposed hand-crafted regularisers (α-norm, total variation) to keep the generated input realistic. Formally, the method generates an inverted representation $\mathbf{x}_i^*$ by

$$\mathbf{x}_i^* = \arg\min_{\tilde{\mathbf{x}}_i} \|\Phi(\mathbf{x}_i) - \Phi(\tilde{\mathbf{x}}_i)\|^2 + \lambda\Omega(\tilde{\mathbf{x}}_i) \tag{1}$$

where $\Omega : \mathbb{R}^n \rightarrow \mathbb{R}$ is the regularisation function. This method although useful has some challenges, one being the handcrafting of the regularisation function. It is hard to define the composition of a naturally occurring input (e.g., image, audio). In order to avoid this, we use the approach from Dosovitskiy et al. [16]. Their method trains a separate deep neural network to invert each layer of D. They claim that the trained model automatically learns a data prior during training. Moreover, their method is computationally expensive only at the training time. Once the network is trained, feature inversion happens in near real time. On the other hand, the method by Mahendran et al. [15] needs to solve Eq. 1 to invert each $\Phi(\mathbf{x}_i)$.

Formally, our method trains an inversion model G[3] that maps a feature vector $\Phi(\mathbf{x}_i)$ to an expected pre-input $\mathbf{x}_i^*$ (weighted average of all the inputs that the forward pass maps to $\Phi(\mathbf{x}_i)$). The method learns the weights $\mathbf{w}^*$ of the inversion model $G(\Phi(\mathbf{x}_i); \mathbf{w})$ by

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_i \|\mathbf{x}_i - G(\Phi(\mathbf{x}_i); \mathbf{w})\|^2 \tag{2}$$

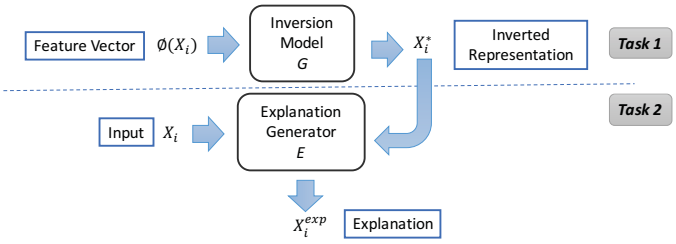

Fig. 1. The functional diagram to depict the explanation generation step of the proposed method. Task 1 involves using an inversion model G to invert a feature vector $\Phi(\mathbf{x}_i)$. Task 2 involves using an explanation generator E and the inverted representation $\mathbf{x}_i^*$ to generate the explanation $\mathbf{x}_i^{exp}$ for the categorisation of the input $\mathbf{x}_i$.

B. Step 2: Explanation Generation

Fig. 1 depicts the tasks involved in the explanation generation step. In order to explain a prediction, task 1 of this step first inverts a feature vector $\Phi(\mathbf{x}_i)$ by

$$\mathbf{x}_i^* = G(\Phi(\mathbf{x}_i); \mathbf{w}^*) \tag{3}$$

Later, the task 2 involves feeding the input excerpt $\mathbf{x}_i$ and its inverted representation $\mathbf{x}_i^*$ to an explanation generator E module that generates an explanation. This module firstly

[3]We can think of G as an approximate inverse of Φ. Φ is a many-to-one function, as several inputs can have the same feature representation. This prevents Φ from having a unique inverse.

ISBN 978-90-827970-1-5 © EURASIP 2018

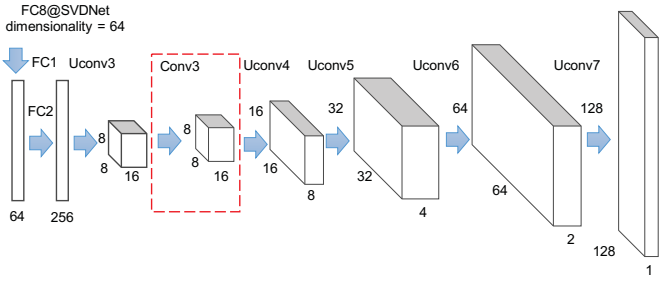

Fig. 2. Model architecture for inverting the FC8 layer of SVDNet [14]. The dotted box encloses the 'Conv3' convolutional layer and its output activation map. Due to space restrictions, we show only one Conv layer.

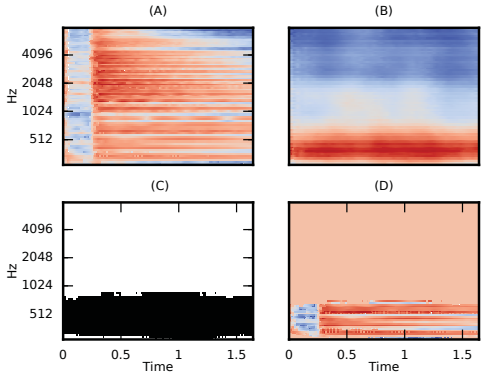

Fig. 3. Local explanation for an excerpt from "03 - Say me Good Bye.mp3" (time index = 10sec) in the Jamendo test dataset. (A) input mel spectrogram, (B) inverted representation, (C) binary mask, and (D) local explanation. All visualisations are in the range [0, 1].

normalises $\mathbf{x}_i^*$ to the range $[0, 1]$ and then thresholds the normalised output using a thresholding constant α_{th}[4] generating a binary mask $\mathbf{x}_i^b \in \{0, 1\}^n$. Later, E generates the explanation $\mathbf{x}_i^{exp}$ for a prediction by masking the input $\mathbf{x}_i$.

$$\mathbf{x}_i^{exp} = \mathbf{x}_i \odot \mathbf{x}_i^b \tag{4}$$

The non-zero (unmasked) bins in $\mathbf{x}_i^{exp}$ highlight the input region maximally influencing the prediction.

III. Experiments

We now demonstrate the proposed method for the singing voice detection (SVD) model introduced in [14]. This model (we refer to as 'SVDNet') classifies an audio excerpt into two categories: instrumental music with and without singing voice. There exist several methods to design efficient SVD models [19], [20]. We choose the method from [14] as the trained model is open-sourced[5] and is state-of-the-art on public benchmark datasets. The model is an eight-layered CNN whose architecture is inspired by VGGNet [21]. The deepest hidden layer in the network is a fully connected layer with 64 neurons (we refer to as 'FC8'). The model uses log-scaled mel spectrogram excerpts of about 1.6sec (115 frames) duration to train its parameters. For more details please refer to [14].

A. Architecture and Training of the Inversion Model

We train an inversion model G to invert the feature vectors from the FC8 layer of SVDNet. The architecture of G is a scaled-down version of the model proposed in [16]. G inverts a 64-dimensional feature vector by systematically upsampling (unpooling) it. We can consider unpooling as an approximate inverse of the pooling operation performed by SVDNet. The network uses "upconvolutional layers" (UConv) [22] to perform unpooling and strided convolution. We also add one convolutional layer (Conv) after every UConv layer as suggested in [23]. This increases the model capacity and generates better visualisations. The network uses batch normalisation layers to force the input to each network layer to follow a standard normal distribution. The network employs Exponential Linear Unit (ELU) non-linearities ($y(x) = (x > 0)?x : e^x - 1$) to process the output of each neuron. G generates an output of 128×128 spatial size and later crops it to 115×80. Fig. 2 depicts an overview of the inversion model architecture and Appendix A provides additional details.

We train G by using 64-dimensional feature vectors extracted from FC8. SVDNet extracts one feature vector per audio excerpt (log-scaled mel spectrogram of about 1.6 sec). We generate audio excerpts from the Jamendo training dataset of 61 pop music songs [24] using a hop size of 10 frames (140 ms). We do not use any of the data augmentation methods from [14]. We train G on a dataset size of about $100k$ feature vectors. In order to prevent overfitting, we use L_2 weight decay and run the optimisation to a fixed number of weight updates (30 epochs). We initialise the model weights using He normal initialisation [25]. For a mini-batch of 32 randomly selected excerpts, the training objective minimises the squared Euclidean distance between the input and its reconstruction and updates the weights using ADAM [26]. Training starts with an initial learning rate of 0.001. If the loss doesn't change for 2 consecutive epochs, the method scales it down by 50%.

B. Local Explanations for SVDNet

We now use the trained inversion model G to explain the predictions of SVDNet as discussed in section II-B. Fig. 3 shows visualisations from the explanation generation step for an excerpt from the Jamendo test dataset. SVDNet correctly classifies this excerpt to 'instrumental music' class with 95.16% confidence. We use G to identify the input region that maximally influences this prediction. Our method normalises and thresholds the inverted representation to generate a binary mask (α_{th} = 0.6). Later, it generates an explanation by applying the mask to the input mel spectrogram. It seems the information in the lower frequency region (till about 900 Hz) maximally influences the prediction of the selected excerpt.

We can use local explanations to verify the trustworthiness of a model. For example, consider the instance shown in Fig.

[4]If a normalised bin value is less than α_{th} then set it to 0 else set it to 1. We use this thresholding method as it seems reasonable to assume that the magnitude of a bin in the inverted representation (expected pre-input) relates to its importance in the discrimination task.

[5]https://github.com/f0k/ismir2015

ISBN 978-90-827970-1-5 © EURASIP 2018

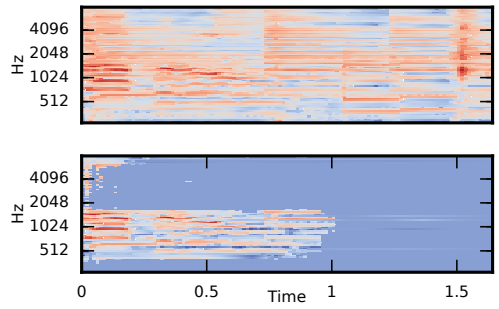

Fig. 4. Local explanation for an excerpt from "03 - Say me Good Bye.mp3" (time index = 34sec) in the Jamendo test dataset with the 'singing voice' and 'instrumental music' classes as separate temporal segments. (Top) input mel spectrogram, and (Bottom) local explanation. All visualisations are in the range [0, 1].

4 that has the singing voice and instrumental music sections located in non-overlapping temporal segments (about 0.9 sec vocals followed by about 0.7 sec instrumental music). SVDNet classifies this instance to the 'singing voice' class with 80% confidence. The local explanation for this prediction shows that frequencies in the range 300 Hz - 1500 Hz in the first one second of the excerpt contribute most to the classifier's prediction. In other words, the model is using information from the vocal segment to categorise the input to the 'singing voice' class. Such an understanding assists in gaining trust in a model's predictions.

C. Quantitative Evaluation of the Proposed Method

Researchers often verify their explanation methods using qualitative approaches (e.g., a method 'A' is better than 'B' if the former has less noisy attribution maps). But, recent works have stressed the importance of quantitative evaluation [27], [28]. In this section, we report the results of quantitative evaluation of our explanation method. To evaluate our method, we adapt the region perturbation method from [27] to suit our region-based explanations. We argue that if an explanation method accurately identifies the influential region in an input, then masking the remaining input should not affect the prediction.[6] Thus, for each input excerpt, our evaluation method feeds the masked representation $\mathbf{x}_i^{exp}$ to SVDNet and records the prediction label.

Formally, for a set of N input excerpts, we consider the initial class labels assigned by SVDNet as the ground truth. We then calculate the number of excerpts (N_E) for which the model prediction changes after input modification.[7] The % explanation loss is given by $E_{loss} = (\frac{N_E}{N}) * 100$. For example, say for $N = 4$ excerpts the SVDNet predictions are $\{0, 1, 1, 0\}$, where 0 and 1 correspond to the 'instrumental music' and 'singing voice' classes, respectively. If after the input modification, the new set of predictions from SVDNet are $\{0, 0, 1, 0\}$. Then, $N_E = 1$ and E_{loss} = 25%.

[6]We mean that the prediction probability might change, but the prediction label should remain the same

[7]We consider initial model predictions irrespective of their class (e.g., singing voice) or their prediction type (e.g., true positive).

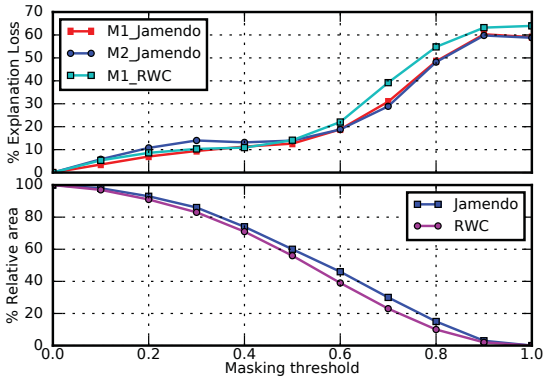

Fig. 5. Quantitative evaluation of the proposed method on a set of randomly selected instances from the Jamendo and RWC datasets. (Top) depicts the change in explanation loss as the input varies from an unmasked version to a fully masked version. (Bottom) shows variations in the relative area of an explanation as we increase the masking threshold α_{th}.

Fig. 5(Top) reports the quantitative evaluation results for a subset of about 3000 randomly chosen excerpts from the Jamendo test dataset of 16 audio files. The plot depicts change in the average E_{loss} with uniform variations in the masking threshold. We evaluate our explanation method for two cases: M1_Jamendo and M2_Jamendo. In M1_Jamendo, the input to SVDNet is the masked representation $\mathbf{x}_i^{exp}$. In M2_Jamendo, we post-process $\mathbf{x}_i^{exp}$ by normalising the masked bins. We normalise a bin using its corresponding mean and standard deviation over the training data. Fig. 5(Top) suggests that both methods report similar explanation loss, especially at higher threshold values. E_{loss} starts at 0% (no input masking) and increases to 60% (full input masking).

Fig. 5(Top) shows that the choice of masking threshold influences the loss resulting from an explanation. Higher values of α_{th} (> 0.7) result in finer but less accurate explanations while lower values of α_{th} (< 0.3) generate highly accurate but coarse explanations. In order to decide an appropriate value of α_{th} we propose to use the relative area of an explanation as an additional constraint. We define the relative area of an explanation as the ratio of the number of unmasked bins to the total number of bins. In Fig. 5(Bottom) we plot the variation in average relative area (for the same excerpts) as α_{th} changes uniformly. We use this information to select a suitable α_{th}. In the above experiment $\alpha_{th} = 0.6$ seems a good choice as it provides finer (% relative area is about 45%) and fairly accurate explanations (% explanation loss is about 20%).

We also extend the evaluation of our explanation method to the RWC dataset [29], which is a public benchmark dataset for SVD. The RWC dataset contains 100 pop music songs and is not pre-split into a separate test set. Thus, we create a test set by randomly sampling 20 audio files and later we randomly sample about 3500 excerpts from the test set we create. We aim to see if the evaluation results from Jamendo generalise to RWC. Fig. 5(Top) depicts the evaluation result for the RWC dataset (using the case 1 approach from Jamendo). For lower values of α_{th} (≤ 0.6), the explanation method performs

ISBN 978-90-827970-1-5 © EURASIP 2018

similary on RWC, but for higher values of α_{th} average E_{loss} increases by about 10%. We argue that it is due to the training of both the discriminator and inversion models only on the Jamendo dataset. This makes the reconstruction error between an input and its inverted representation more for RWC leading to less accurate explanations. Thus, an accurate inversion model is crucial to achieving low explanation loss.

IV. Conclusions and Future Work

In this work, we proposed a novel method to explain predictions of deep machine listening systems. Our method uses the idea of feature inversion to generate local explanations. The method inverts a feature vector from the deepest layer of a model and uses the inverted representation to generate a binary mask to mask the input time-frequency representation. The unmasked portion of the input highlights the region most influential to a prediction. We demonstrated the proposed method for a state-of-the-art SVD model. We used the explanations to analyse the trustworthiness of the model. Finally, we evaluated the loss associated the explanation method. The evaluation results for two benchmark datasets suggested that explanations from the proposed method highlighted the most influential region in about 82% of the randomly chosen inputs.

In the future, we plan to apply our method to a multi-class classification task (e.g., sound event detection). We also plan to compare our method's performance against other region-based explanation methods. Moreover, we also plan to experiment with other thresholding techniques (e.g., using reconstruction error values) to understand the effect of a thresholding technique on the performance of the method.

Acknowledgment

The authors would like to thank Jan Schlüter for discussions and sharing the SVD model. We also thank the anonymous reviewers for their valuable comments and suggestions.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing Properties of Neural Networks," in *Proc. ICLR*, 2014.

[3] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," in *Proc. ECCV*, 2014.

[4] G. Montavon, W. Samek, and K.-R. Müller, "Methods for Interpreting and Understanding Deep Neural Networks," *Digital Signal Processing : A Review Journal*, vol. 73, pp. 1–15, 2018.

[5] C. Olah, A. Mordvintsev, and L. Schubert, "Feature Visualization," *Distill*, 2017.

[6] A. Nguyen, J.Yosinski, and J. Clune, "Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned By Each Neuron in Deep Neural Networks," in *Proc. ICML Workshop*, 2016.

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

[8] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why Should I Trust You?": Explaining the Predictions of Any Classifier," in *Proc. KDD*, 2016.

[9] K. Choi, G. Fazekas, and M. B. Sandler, "Explaining Deep Convolutional Neural Networks on Music Classification," *arXiv e-prints*, vol. arXiv:1607.02444, 2016.

[10] S. Mishra, B. L. Sturm, and S. Dixon, "Local Interpretable Model-Agnostic Explanations for Music Content Analysis," in *Proc. ISMIR*, 2017.

[11] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, "Smoothgrad: Removing Noise by Adding Noise," in *Proc. ICML Workshop on Visualisation for Deep Learning*, 2017.

[12] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for Simplicity: The All Convolutional Net," in *Proc. ICLR Workshop*, 2015.

[13] P. Kindermans, S. Hooker, J. Adebayo, M. Alber, K. T. Schütt, S. Dähne, D. Erhan, and B. Kim, "The (Un)reliability of Saliency Methods," in *Proc. NIPS Workshop*, 2017.

[14] J. Schlüter and T. Grill, "Exploring Data Augmentation for Improved Singing Voice Detection with Neural Networks," in *Proc. ISMIR*, 2015.

[15] A. Mahendran and A. Vedaldi, "Understanding Deep Image Representations by Inverting Them," in *Proc. CVPR*, 2015.

[16] A. Dosovitskiy and T. Brox, "Inverting Visual Representations with Convolutional Networks," in *Proc. CVPR*, 2016.

[17] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet Classification with Deep Convolutional Neural Networks," in *Proc. NIPS*, 2012.

[18] A. Nguyen, J.Yosinski, and J. Clune, "Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images," in *Proc. CVPR*, 2015.

[19] B. Lehner, G. Widmer, and R. Sonnleitner, "On the Reduction of False Positives in Singing Voice Detection," in *Proc. ICASSP*, 2014.

[20] S. Leglaive, R. Hennequin, and R. Badeau, "Singing Voice Detection with Deep Recurrent Neural Networks," in *Proc. ICASSP*, 2015.

[21] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *Proc. ICLR*, 2015.

[22] A. Dosovitskiy, J. T. Springenberg, and T. Brox, "Learning to Generate Chairs with Convolutional Neural Networks," in *Proc. CVPR*, 2015.

[23] A. Dosovitskiy, J. T. Springenberg, M. Tatarchenko, and T. Brox, "Learning to Generate Chairs, Tables and Cars with Convolutional Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[24] M. Ramona, G. Richard, and B. David, "Vocal Detection in Music Using Support Vector Machines," in *Proc. ICASSP*, 2008, pp. 1885–1888.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," in *Proc. ICCV*, 2015.

[26] D. Kingma and B. Jimmy, "Adam: A Method for Stochastic Optimization," in *Proc. ICLR*, 2015.

[27] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K.-R. Müller, "Evaluating the visualization of what a deep neural network has learned," *IEEE Transactions on Neural Networks and Learning Systems*, 2016.

[28] F. Doshi-Velez and B. Kim, "Towards a Rigorous Science of Interpretable Machine Learning," *arXiv e-prints*, vol. arXiv:1702.08608, 2017.

[29] 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.

Appendix

TABLE I

THE ARCHITECTURE OF THE INVERSION MODEL TO INVERT THE FC8 LAYER OF SVDNET. INPUT AND OUTPUT SHAPES ARE ORDERED AS: NUMBER OF CHANNELS × TIME × FREQUENCY. FILTERS IN THE UCONV AND CONV LAYERS ARE OF SHAPES 4×4 AND 3×3, RESPECTIVELY. STRIDE SIZE FOR CONVOLUTIONS IN UCONV AND CONV LAYERS IS 2×2 AND 1×1, RESPECTIVELY. THE NUMBER OF LEARNABLE PARAMETERS FOR THIS NETWORK IS 31887.

Layer	Input Shape	Number of Filters	Output Shape
FC1	64×1	64	64×1
FC2	64×1	256	256×1
Reshape	256×1	-	$16 \times 4 \times 4$
UConv3	$16 \times 4 \times 4$	16	$16 \times 8 \times 8$
Conv3	$16 \times 8 \times 8$	16	$16 \times 8 \times 8$
UConv4	$16 \times 8 \times 8$	8	$8 \times 16 \times 16$
Conv4	$8 \times 16 \times 16$	8	$8 \times 16 \times 16$
UConv5	$8 \times 16 \times 16$	4	$4 \times 32 \times 32$
Conv5	$4 \times 32 \times 32$	4	$4 \times 32 \times 32$
UConv6	$4 \times 32 \times 32$	2	$2 \times 64 \times 64$
Conv6	$2 \times 64 \times 64$	2	$2 \times 64 \times 64$
UConv7	$2 \times 64 \times 64$	1	$1 \times 128 \times 128$