Deep Learning and Music Adversaries

Corey Kereliuk, Bob L. Sturm, *Member, IEEE*, and Jan Larsen, *Senior Member, IEEE*

***Abstract*—An *adversary* is an agent designed to make a classification system perform in some particular way, e.g., increase the probability of a false negative. Recent work builds adversaries for deep learning systems applied to image object recognition, exploiting the parameters of the system to find the minimal perturbation of the input image such that the system misclassifies it with high confidence. We adapt this approach to construct and deploy an adversary of deep learning systems applied to music content analysis. In our case, however, the system inputs are magnitude spectral frames, which require special care in order to produce valid input audio signals from network-derived perturbations. For two different train-test partitionings of two benchmark datasets, and two different architectures, we find that this adversary is very effective. We find that convolutional architectures are more robust compared to systems based on a majority vote over individually classified audio frames. Furthermore, we experiment with a new system that integrates an adversary into the training loop, but do not find that this improves the resilience of the system to new adversaries.**

***Index Terms*—AEA-MIR content-based processing and music information retrieval, deep learning.**

I. Introduction

DEEP learning has impacted the research domain of music content analysis and music information retrieval (MIR) [1]–[9], but recent developments raise the spectre that the high performance of these systems does not reflect how well they have learned to solve high-level problems [10]–[16]. MIR aims to produce systems that help make "music, or information about music, easier to find" [17]. This is of principal importance when confronting the vast amount of music data that exists and continues to be created. Listening machines that can produce accurate, meaningful and searchable descriptions of music can greatly reduce the cost of processing music data, and can facilitate a diversity of applications. These extend from music identification [18], author attribution [19], recommendation [5], transcription [20], and playlist generation [21], to extracting semantic descriptors such as genre and mood [12], [22], [23], to computational musicology [24], and even synthesis and music composition [25].

Recent surveys of the domain of deep learning include impressive results for several benchmark problems [26], [27]. In addition to these major successes, deep learning methods are very attractive for three other reasons: there now exist efficient and effective training algorithms, not to mention completely free and open cross-platform implementations, e.g., *Theano* [28], [29]; they entail jointly optimising feature learning and classification, thus allowing one to forgo many difficulties inherent to formally encoding expert knowledge into a machine; and their layered structures seem to favour hierarchical representations of structures in data. A caveat is that these methods require a lot of data in order to estimate parameters and generalise well [30], and furthermore a suitable set of hyperparameters must be selected (of which there may be many).

In MIR, the works in [1], [2], [31] are among the first to apply deep learning to music content analysis, and each describe results concluding that these systems can automatically learn features relevant for complex music listening tasks, e.g., recognition of genre or style. Results since then point to the same conclusion [3], [6]–[9]. Humphrey *et al.* [4] highlight this fact to argue deep learning is naturally suited to learn relevant abstractions for music content analysis, provided enough data is available. Since music may be seen as a "whole greater than the sum of its parts" [4], deep learning can help MIR narrow the "semantic gap" [32], and move beyond what has been called a "glass ceiling" in performance [33].

However, the appearance of high performance is sometimes deceiving: an MIR system can appear to be very successful in solving a high-level music listening problem when in fact it is just exploiting some independent variables of questionable relevance confounded with the ground truth of a dataset [11], [12], [13], [34]–[40]. In addition, recent work has demonstrated deep learning systems behaving in ways that contradict their appearance of solving content-recognition problems. Nguyen *et al.* [41] show how a state of the art image object recognition system labels non-sensical synthetic images with high-confidence. In a similar direction, we have shown [39] how a system that appears highly capable of recognising different musical rhythms confidently classifies synthesised rhythms that bear little similarity to the rhythms they supposedly represent. Szegedy *et al.* [10] show how state of the art image object recognition systems are highly sensitive to imperceptible perturbations created by an *adversary*: an agent that actively seeks to fool a classifier by perturbing the input such that it results in an incorrect output with high confidence [42].

Manuscript received March 15, 2015; revised July 08, 2015; accepted August 26, 2015. Date of publication September 10, 2015; date of current version October 20, 2015. The work of C. Kereliuk and J. Larsen was supported in part by the Danish Council for Strategic Research, Danish Agency for Science Technology and Innovation under the CoSound project, Case 11-115328. The guest editor coordinating the review of this manuscript and approving it for publication was Dr. Hugo Larochelle.

C. Kereliuk is with DTU Compute, The Technical University of Denmark, Frederiksber 4TV2000, Denmark (e-mail: coreyker@gmail.com).

B. L. Sturm is with the School of Electronic Engineering and Computer Science, Queen Mary University of London, London E1 4NS, U.K (e-mail: b.sturm@qmul.ac.uk).

J. Larsen is with the Technical University of Denmark, Kongens Lyngby DK-2800, Denmark (e-mail: janla@dtu.dk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2015.2478068

All of these results motivate several questions of deep learning systems for music content analysis. First, how do the adversaries of Szegedy *et al.* [10] translate to the context of deep learning applied to music content analysis? The input of the systems studied by Szegedy *et al.* [10] is raw pixel data; however, in music content analysis, only the system studied in [8] takes as input raw PCM (time-domain) audio samples. The inputs to other systems have been features (all of which discard information before it has the chance to reach the network): windowed magnitude spectra [2], [7], sonograms [1], [5], autocorrelations of spectral energies [6], [39], or statistics of features [3], [9]. Second, can we generate an adversary for such deep learning-based music content analysis systems that produce adversarial examples that are perceptually identical to the originals? Third, can we "harness" an adversary to train deep learning systems that are robust to its "malfeasance"? Finally, and more broadly, what is deep learning contributing to music content analysis? Can we use adversaries to reveal whether these systems have learned better models of the content than other state of the art systems using hand-crafted features?

Our preliminary work [14] shows we can create highly effective adversaries of the music content analysis deep neural networks (DNN) studied in [2], [7]. These adversaries can make the systems always wrong, always right, and anywhere in-between, with high confidence through the application of minor perturbations to the input magnitude spectra. Furthermore, we created an ensemble of adversaries that can coax the DNN into assigning with high confidence any label to the same music by perturbing the input by very small amounts (e.g., 26.8 dB SNR). In this article, we expand upon prior work [14] to include convolutional deep learning systems, extensive testing in a larger benchmark MIR dataset, and the results of incorporating an adversary into the training of deep learning systems.

In the next section, we provide an overview of work applying deep learning to music content analysis and MIR. We then review two different deep learning architectures, and our construction of several music content analysis systems using two partitions of two MIR benchmark datasets. In Section III we review adversaries, and design an adversary for our deep systems. We then present in Section IV a series of experiments using our adversary. In Section V we provide a discussion of our work in wider contexts. We conclude in Section VI. Some of our results can be produced with the software here: https://github.com/coreyker/dnn-mgr.

II. Deep Learning for Music Content Analysis

We first provide an overview of research applying deep learning approaches to music content analysis. We then discuss two different architectures, train two music content analysis systems, and test them in two benchmark MIR datasets. These systems are the subjects of our experiments in Section IV.

A. Overview

Artificial neural networks have been applied to many music content analysis problems [43], for instance, fingerprinting [44], genre recognition [45], emotion recognition [46], artist recognition [47], and even composition [48]. Advances in training algorithms have enabled the creation of more advanced and deeper architectures. Deng and Yu [26] (Chapter 7) provide a review of successful applications of deep learning to the analysis of audio, highlighting in particular its significant contributions to speech recognition in conversational settings. Humphrey *et al.* [4] provide a review for applications to music in particular, and motivate the capacity of deep architectures to automatically learn hierarchical relationships in accordance with the hierarchical nature of music: "pitch and loudness combine over time to form chords, melodies and rhythms." They argue that this is key for moving beyond the reliance on "shallow" and hand-designed features that were designed for different tasks.

Lee *et al.* [1] are perhaps the first to apply deep learning to music content analysis, specifically genre and artist recognition. They train a convolutional deep belief network (CDBN) with two hidden layers in an unsupervised manner in an attempt to make the hidden layer activations produce meaningful features from a pre-processed spectrogram input computed using 20 ms 50%-overlapped windows. The spectrogram is "PCA-whitened", which involves projecting it onto a lower-dimensional space using scaled eigenvectors. Details are missing in the description of their work, but it appears they use the activations as features in some train/test task using a standard machine learning approach. A table of their experimental results, using some portion of the dataset *ISMIR2004*, shows higher accuracies for their deep learned features compared to those for standard MFCCs. For genre recognition, Li *et al.* [31] use convolutional deep neural networks (CDNN) with three hidden layers, into which they input a sequence of 190 13-dimensional MFCC feature vectors. The architecture of their CDNN is such that the first hidden layer considers data from 127 ms duration, and the last hidden layer is capable of summarising events over a 2.2 s duration. van den Oord *et al.* [5] apply CDNN to mel-frequency spectrograms for automatic music content analysis, and ultimately music recommendation.

For genre recognition and more general descriptors, Hamel and Eck [2] train a DNN with three hidden layers of 50 units each, taking as input 513 discrete Fourier transform (DFT) magnitudes computed from a single 46 ms audio frame. They use a train/valid/test partition of the benchmark music genre dataset *GTZAN* [12], [49]. They also explore "aggregated" features, which are the mean and variance in each dimension of activations over 5 second durations. They find for both short-term and aggregated features that SVM classifiers trained with features built from hidden layer activations reproduce more ground truth than an SVM classifier trained with features built from MFCCs. They report an accuracy of over 0.84 for features that aggregate activations of all three hidden layers. Sigtia and Dixon [7] explore modifications to the system in [2], in particular using different combinations of architectures, training procedures, and regularisation. They use the activations of their trained DNN as features for a train/test task using a random forest classifier. They report an accuracy of about 0.83 using features aggregating activations across all hidden layers of 500 units each. For genre recognition, Yang *et al.* [3] combine 263-dimensional modulation features with a DBN. For music rhythm classification, Pikrakis [6] employs a DBN, which we study further in [13], [39], [40].

Dieleman *et al.* [50] build and apply CDBN to music key detection, artist recognition, and genre recognition. There are three

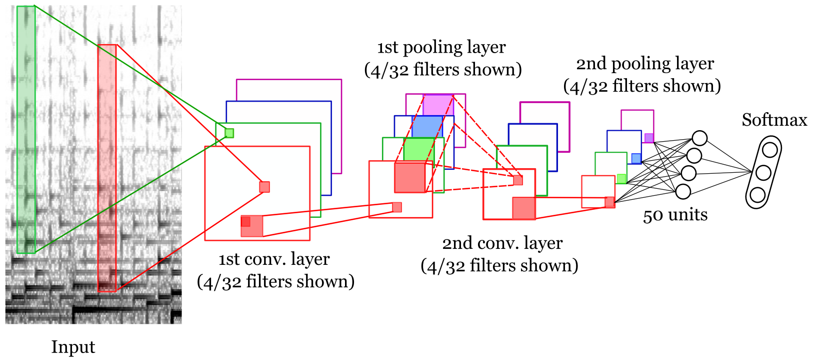


Fig. 1. Illustration of the CDNN architecture we use for our experiments. The CDNN first applies narrow vertical filters to the input sonogram (left) to capture the harmonic structure (we show only four of 32). The convolutional layer is cascaded with a max-pooling layer which downsamples the output. This is followed by a second pair of convolutional and max-pooling layers. Finally, the output of the final max-pooling layer is fully connected to a final hidden layer of 50 units, followed by a softmax output unit (giving posterior probabilities of over the set of class labels). The input spectrogram contains 100 time slices sampled every 23 ms, which means that the final layer of the CDNN summarizes information over a total duration of 2.3 seconds.

major differences with respect to the work above [1]–[3], [7], [31]. First, Dieleman *et al.* employ 24-dimensional input features computed by averaging short-time chroma and timbre features over the time scales of single musical beats. Second, they employ expert musical knowledge to guide decisions about the architecture of the system. Finally, they use the output posteriors of their system for classification, instead of using the hidden layer activations as features for a separate classifier. Their experiments in a portion of the "million song dataset" [51] show large differences in classification accuracies between their systems and a naive Bayesian classifier using the same input features. In a unique direction for audio, Dieleman and Schrauwen [8] explore "end-to-end" learning, where a CDNN is trained with input of about 3 s of raw audio samples for a music content analysis task (autotagging). They find that the lowest layer of the trained CDNN appears to learn some filters that are frequency selective. They evaluate this system for a multilabel problem.

To recognize music mood, Weninger *et al.* [52] use a recurrent DNN with inputs constructed from several statistics of low-level features computed over second-long excerpts of music recordings. Battenberg and Wessel [53] apply DBN for identifying the beat numbers over several measures of percussive music, with input features consisting of quantised onset times and magnitudes. Boulanger-Lewandowski *et al.* [54] train a recurrent neural network to produce chord classifications using inputs from PCA-whitened magnitude DFT. In a similar direction, Humphrey and Bello [55] build a DNN that maps input spectrogram features to guitar-specific fingerings of chords.

B. Two Types of Deep Architectures

We now review two different architectures of deep learning systems, and the way they are trained. A DNN is an artificial neural network with several hidden layers [26]. The output of each layer is a non-linear function of its inputs, obtained by a matrix multiplication cascaded with a non-linearity, e.g., tanh, sigmoid and rectifier. By chaining together several hidden layers, composite representations of the input emerge in deeper layers. This fact can give deep networks greater representational power than shallower networks containing an equivalent number of parameters [56].

A CDNN is a special type of DNN with weights that are shared between multiple points between adjacent layers. The weight sharing in CDNNs not only reduces the number of trainable parameters, but also causes matrix multiplications to reduce to convolutions, which can be implemented efficiently. Furthermore, many natural signals have local spatial or temporal structures that are repeated globally. For example, natural images often consist of oriented edges; and music audio signals often consist of time-frequency structures. CDNNs appear able to learn these types of structures very well. Fig. 1 illustrates our CDNN, which we discuss in the following subsection.

The contemporary success of deep learning comes with computationally efficient training methods. Systems that have such deep architectures are usually trained using gradient descent, which consists of backpropagating error derivatives from the cost function through the network. There are a plethora of useful tips and tricks to augment training, including stochastic gradient descent, dropout regularization, weight decay, momentum, learning rate decay, and so on [30].

C. Deep Learning With Two Music Genre Benchmarks

We build our deep systems using two music genre benchmarks: *GTZAN* [12], [49] and the Latin Music Database (*LMD*) [57]. *GTZAN* consists of 100, 30-second music excerpts in each of ten categories, and is the most-used public dataset in MIR research [58]. *LMD* is a private dataset, consisting of 3,229 full-length music recordings non-uniformly distributed among ten categories, and has been used in the annual MIREX audio Latin music genre classification evaluation campaign since 2008.[1] We use the first 30 seconds of each track in *LMD*.

We use different partitionings of these datasets: One partitioning of *GTZAN* we create by randomly selecting 500/250/250 excerpts for training/validation/testing. The other partitioning of *GTZAN* is "fault-filtered," which we construct by hand to include 443/197/290 excerpts. This involves removing 70 files including exact replicas, recording replicas, and distorted files [12], and then dividing the excerpts such that no artist is repeated across the training, validation, and test partitions. We partition *LMD* in two ways: 1) partitioning 60/20/20% in each

[1][Online]. Available: http://www.music-ir.org/mirex/wiki/MIREX_HOME

class randomly; and 2) a hand-constructed artist-filtered partitioning containing approximately the same division of excerpts in each category. We retain all 213 replicas in *LMD*.[2]

The input to our systems is derived from the short-time Fourier transform (STFT) of a sampled audio signal x [59]

$$\mathcal{F}(x)[m,u] = \sum_{l=0}^{L-1} w[l]x[l-uH]e^{-j2\pi ml/L} \tag{1}$$

where the parameter L defines both the window length and the number of frequency bins. We define w as a Hann window of length $L = 1024$, which corresponds to a duration of 46 ms for recordings sampled at 22050 Hz. The window is hopped along x with a stride of $H = 512$ samples (adjacent windows overlap by 50%).

Since audio signals can be of any duration, we define the input to our systems as a sequence $X = (X_n)_{n=0}^{N-1}$, where the sequence length depends on the input audio's duration. We define the nth element of the input sequence X to be

$$X_n \triangleq (|\mathcal{F}(x)[m,u]| : m \in [0,512], u \in [nT,(n+1)T]) \tag{2}$$

where $T = 1$ for each DNN and $T = 100$ for each CDNN. Thus, when $T = 1$, X is a sequence of 513-dimensional vectors; when $T = 100$, X is a sequence of matrices of size 513×100.

Our systems process each element in this sequence independently, outputting a sequence $P = (P_n)_{n=0}^{N-1}$ from the final (softmax) layer. The output vector $P_n \in 0,1^K$, $\|P_n\|_1 = 1$, is the posterior distribution of labels assigned to the nth element in the input sequence by the network. Therefore, we may write $P_n(i|X_n,\Theta) \equiv P_n(i) \in [0,1]$ where Θ represents the trainable network parameters, i.e., the set of weights and biases. We define the *confidence* of a system in a particular label $k \in \{1,\ldots,K\}$ for an input sequence X as the sum of all posteriors, i.e.

$$R(k|X,\Theta) = \frac{1}{N}\sum_{n=0}^{N-1} P_n(k|X_n,\Theta). \tag{3}$$

We apply a label to an input sequence X as the one maximizing the confidence

$$y(X,\Theta) = \arg\max_{k\in\{1,\ldots,K\}} R(k|X,\Theta). \tag{4}$$

Paralleling the work in [7], we build DNNs with 3 fully connected hidden layers, and either 50 or 500 units per layer. Our CDNN has two convolutional layers (accompanied by max pooling layers) followed by a fully connected hidden layer with 50 units. Fig. 1 illustrates the architecture of our CDNN. Its first convolutional layer contains 32 filters, encompassing 400 frequency bins and 4 time slices. We choose this long rectangular shape instead of the small square patches typically used when training on images based on our knowledge that many sounds exhibit strong harmonic structures that span a large portion of the audible spectrum. The second convolutional layer contains 32 filters, each connected in an 8×8 pattern. Our two pooling neighborhoods are 4×4 and have strides of 2×2. All of our deep learning systems use rectified linear units (ReLUs), and have a softmax unit in the final layer. As is typical, we standardize the inputs to our systems by subtracting the training set mean and dividing by the standard deviation in each of the input dimensions. We perform this with a linear layer above the input layer of each network.

As done in [7], we build several music classification systems treating our DNN as a feature extractor. In this case, we construct a set of features by concatenating the activations from the three hidden layers, and aggregating them over 5-second texture windows (hopped by 50%). The aggregation summarizes the mean and standard deviation of the feature dimensions over the texture window and may be seen as a form of late-integration of temporal information. We use this new set of features to train a random forest (RF) classifier [60] with 500 trees. To classify a music audio recording x from its set of aggregated features, we use majority voting over all classifications (also used in [7]).

D. Preliminary Evaluation

Fig. 2 and Table I show the results of RF classification using the features produced by the DNN when trained on *GTZAN* with the two different partitioning strategies; and Fig. 3 shows those for the (C)DNNs we train and test in *LMD*. Across each partition strategy we see significant differences in performance. The mean recall in each class in Fig. 2 on the fault-filtered partition is much lower than that on the random test partition—involving drops higher than 30 percentage points in most cases. Table I shows similar drops in performance that persist over the inclusion of drop-out regularisation. Such significant drops in performance from artist-based partitioning is not unusual, and has been studied before as a bias coming from the experimental design [12], [34], [36]. Partitioning a music genre recognition dataset along artist lines has been recommended to avoid this bias [34], [36], and is in fact used in several MIREX audio classification tasks.[3] Experiments using *GTZAN* with fault-filtering partitioning has not been used in many benchmark experiments with *GTZAN* because its artist information was, until recently, unknown [12].

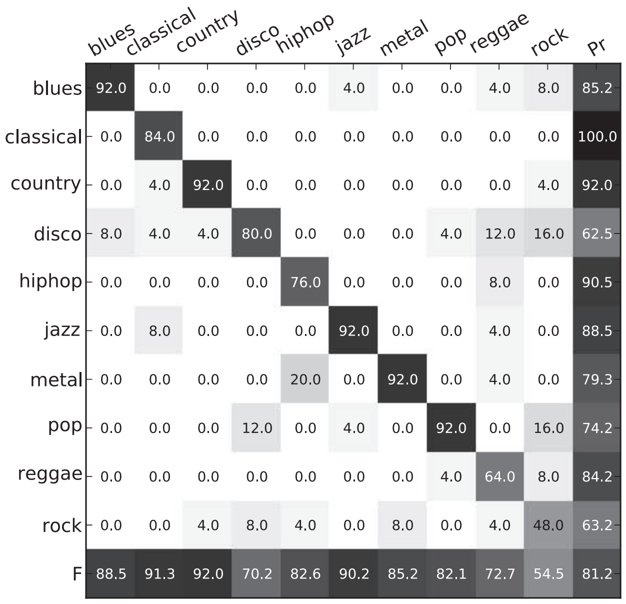
III. Adversaries in Music Content Analysis

An adversary is an agent designed to fool a classification system in order to maximise some gain, e.g., defeating SPAM detection. Dalvi *et al.* [42] pose this problem as a game between a classifier and adversary, and analyze the strategies involved for an adversary with complete knowledge of the classification system, and for a classifier to adapt to such an adversary. Szegedy *et al.* [10] propose using adversaries for testing the assumption that deep learning systems are "smooth classifiers," i.e., stable in their classification to small perturbations around examples in the training data. They define an adversary of a classifier $f : \mathbb{R}^m \rightarrow \{1,\ldots,K\}$ as an algorithm using complete knowledge of the classifier to perturb an observation $x \in \mathbb{R}^m$ such that $f(x+r) \neq f(x)$, where $r \in \mathbb{R}^m$ is some perturbation. Specifically, their adversary solves the constrained optimization problem for a given $k \in \{1,\ldots,K\}$

$$\min \|r\|_2 \text{subject to } f(x+r) = k. \tag{5}$$

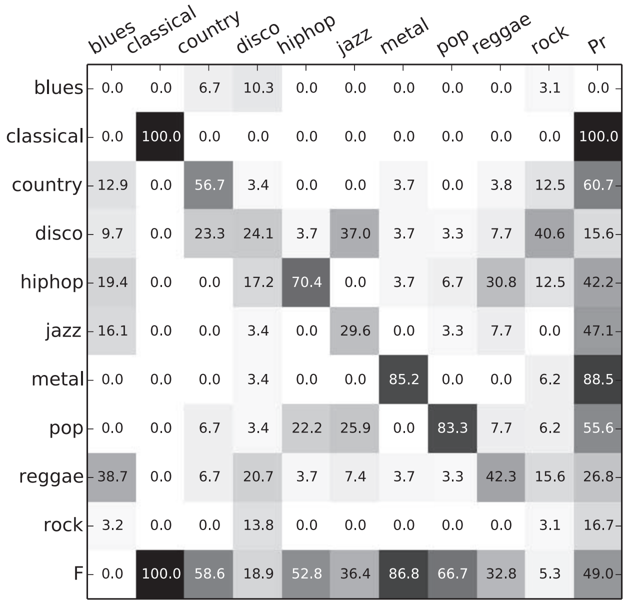
[2][Online]. Available: https://highnoongmt.wordpress.com/2014/02/08/faults_in_the_latin_music_database

[3][Online]. Available: http://www.music-ir.org/mirex/wiki/MIREX_HOME



	blues	classical	country	disco	hiphop	jazz	metal	pop	reggae	rock	Pr
blues	92.0	0.0	0.0	0.0	0.0	4.0	0.0	0.0	4.0	8.0	85.2
classical	0.0	84.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
country	0.0	4.0	92.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0	92.0
disco	8.0	4.0	4.0	80.0	0.0	0.0	0.0	4.0	12.0	16.0	62.5
hiphop	0.0	0.0	0.0	0.0	76.0	0.0	0.0	0.0	8.0	0.0	90.5
jazz	0.0	8.0	0.0	0.0	0.0	92.0	0.0	0.0	4.0	0.0	88.5
metal	0.0	0.0	0.0	0.0	20.0	0.0	92.0	0.0	4.0	0.0	79.3
pop	0.0	0.0	0.0	12.0	0.0	4.0	0.0	92.0	0.0	16.0	74.2
reggae	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0	64.0	8.0	84.2
rock	0.0	0.0	4.0	8.0	4.0	0.0	8.0	0.0	4.0	48.0	63.2
F	88.5	91.3	92.0	70.2	82.6	90.2	85.2	82.1	72.7	54.5	81.2

(a)



	blues	classical	country	disco	hiphop	jazz	metal	pop	reggae	rock	Pr
blues	0.0	0.0	6.7	10.3	0.0	0.0	0.0	0.0	0.0	3.1	0.0
classical	0.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
country	12.9	0.0	56.7	3.4	0.0	0.0	3.7	0.0	3.8	12.5	60.7
disco	9.7	0.0	23.3	24.1	3.7	37.0	3.7	3.3	7.7	40.6	15.6
hiphop	19.4	0.0	0.0	17.2	70.4	0.0	3.7	6.7	30.8	12.5	42.2
jazz	16.1	0.0	0.0	3.4	0.0	29.6	0.0	3.3	7.7	0.0	47.1
metal	0.0	0.0	0.0	3.4	0.0	0.0	85.2	0.0	0.0	6.2	88.5
pop	0.0	0.0	6.7	3.4	22.2	25.9	0.0	83.3	7.7	6.2	55.6
reggae	38.7	0.0	6.7	20.7	3.7	7.4	3.7	3.3	42.3	15.6	26.8
rock	3.2	0.0	0.0	13.8	0.0	0.0	0.0	0.0	0.0	3.1	16.7
F	0.0	100.0	58.6	18.9	52.8	36.4	86.8	66.7	32.8	5.3	49.0

(b)

Fig. 2. Figure of merit (FoM, $\times 100$) in *GTZAN* with two different partitionings for random forest classification (majority vote) of DNN-based features (all layers) aggregated over 5-second windows. Each DNN uses 500 rectified linear units in each hidden layer. Columns represent the dataset ground truth; rows denote labels chosen by system; the diagonal contains the per-class recall; the off-diagonal entries are confusions; the rightmost column is the precision; the bottom row is the F-score; and the last element along the diagonal is the mean recall. (a) Random partitioning. (b) Fault-filtered partitioning.

TABLE I
RECALL ($\times 100$) IN *GTZAN* FOR RANDOM FOREST CLASSIFICATION OF DNN-BASED FEATURES AGGREGATED OVER 5-SECOND WINDOWS. THE NUMBER OUTSIDE BRACKETS IS FROM THE RANDOM PARTITION, AND THAT INSIDE BRACKETS IS FROM THE FAULT-FILTERED PARTITION

Hidden Units	Layer	ReLU	ReLU+Dropout
50	1	76.00 (40.69)	80.40 (45.17)
	2	78.80 (45.17)	80.40 (43.10)
	3	79.60 (43.79)	78.80 (44.48)
	All	80.40 (43.79)	80.00 (43.79)
500	1	68.40 (40.34)	75.60 (40.69)
	2	74.40 (40.69)	80.00 (50.34)
	3	77.60 (43.79)	79.20 (48.62)
	All	76.00 (42.41)	81.20 (48.97)

For $k \neq f(x)$, Szegedy *et al.* [10] employ a line search along the direction of the loss function of the network starting from x until the classifier produces the requested class. They find that adversarial examples of one classifier can fool other classifiers trained on independent data; hence, one need not have complete knowledge of a classifier in order to fool it.

Goodfellow *et al.* [15] provide an intuitive explanation of these adversaries: even though the perturbations in each dimension might be small, their contribution to the magnitude of a projection grows linearly with input dimensionality. With a deep neural network involving many such projections in each layer, a small perturbation at its high-dimensional input layer can create major consequences at the output layer. Goodfellow *et al.* [15] show that adversarial examples can be easily generated by making the perturbation proportional to the sign of the partial derivative of the loss function used to train the network, evaluated with respect to the class the adversary wishes to minimize. They also find that the direction of perturbation is important, not necessarily its size. Hence, it seems adversarial examples of one model will likely fool other models because they occur in large volumes in high-dimensional spaces. This is also found by Gu and Rigazio [16].

As for Szegedy *et al.* [10], we are interested the robustness of our deep learning systems to an adversary. Do our systems suffer just as dramatically as the image content recognition systems in [10], [15], [16]? In other words, can we find imperceptible perturbations of music recordings, yet make the systems produce any label with high confidence? If so, can we adapt the training of the systems such that they become more robust? In the next subsections, we define an adversary as an optimization problem, but with care of the fact that the input to our deep learning systems are magnitude STFT (2). We then present an approach to integrate adversaries into the training of our systems. We present our experimental results in Section IV.

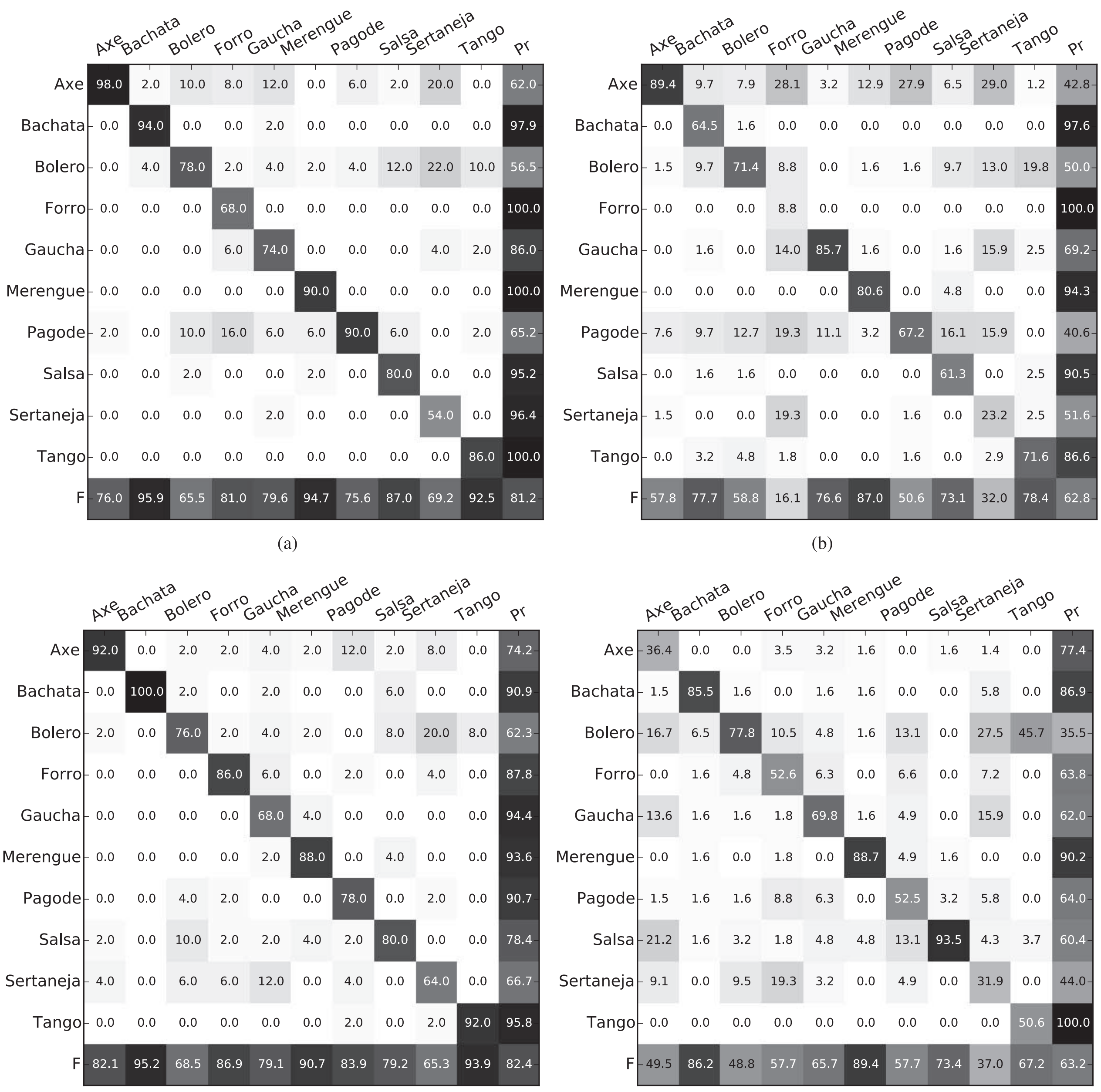
A. Music Adversaries

The explicit goal of our adversary is to perturb a music recording x such that a system will confidently classify it with some class $y \in \{1, \ldots, K\}$. Specifically, we define the adversary as the constrained optimization problem

$$\hat{X}(y) = \arg \min_{Z \in C(X)} \sum_{n=0}^{N-1} \mathcal{L}(Z_n, y|\Theta) \tag{6}$$

where we define the feasible set of adversarial examples to input sequence X as

$$\mathcal{C}(X) = \left\{ Z = (Z_n)_{n=0}^{N-1} : \sqrt{\sum\nolimits_{n=0}^{N-1} \|Z_n - X_n\|_2^2} \right. \\ \left. \leq N\epsilon(\text{SNR}) \right\} \tag{7}$$



	Axe	Bachata	Bolero	Forro	Gaucha	Merengue	Pagode	Salsa	Sertaneja	Tango	Pr
Axe	98.0	2.0	10.0	8.0	12.0	0.0	6.0	2.0	20.0	0.0	62.0
Bachata	0.0	94.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	97.9
Bolero	0.0	4.0	78.0	2.0	4.0	2.0	4.0	12.0	22.0	10.0	56.5
Forro	0.0	0.0	0.0	68.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0
Gaucha	0.0	0.0	0.0	6.0	74.0	0.0	0.0	0.0	4.0	2.0	86.0
Merengue	0.0	0.0	0.0	0.0	0.0	90.0	0.0	0.0	0.0	0.0	100.0
Pagode	2.0	0.0	10.0	16.0	6.0	6.0	90.0	6.0	0.0	2.0	65.2
Salsa	0.0	0.0	2.0	0.0	0.0	2.0	0.0	80.0	0.0	0.0	95.2
Sertaneja	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	54.0	0.0	96.4
Tango	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	86.0	100.0
F	76.0	95.9	65.5	81.0	79.6	94.7	75.6	87.0	69.2	92.5	81.2

(a)

	Axe	Bachata	Bolero	Forro	Gaucha	Merengue	Pagode	Salsa	Sertaneja	Tango	Pr
Axe	89.4	9.7	7.9	28.1	3.2	12.9	27.9	6.5	29.0	1.2	42.8
Bachata	0.0	64.5	1.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	97.6
Bolero	1.5	9.7	71.4	8.8	0.0	1.6	1.6	9.7	13.0	19.8	50.0
Forro	0.0	0.0	0.0	8.8	0.0	0.0	0.0	0.0	0.0	0.0	100.0
Gaucha	0.0	1.6	0.0	14.0	85.7	1.6	0.0	1.6	15.9	2.5	69.2
Merengue	0.0	0.0	0.0	0.0	0.0	80.6	0.0	4.8	0.0	0.0	94.3
Pagode	7.6	9.7	12.7	19.3	11.1	3.2	67.2	16.1	15.9	0.0	40.6
Salsa	0.0	1.6	1.6	0.0	0.0	0.0	0.0	61.3	0.0	2.5	90.5
Sertaneja	1.5	0.0	0.0	19.3	0.0	0.0	1.6	0.0	23.2	2.5	51.6
Tango	0.0	3.2	4.8	1.8	0.0	0.0	1.6	0.0	2.9	71.6	86.6
F	57.8	77.7	58.8	16.1	76.6	87.0	50.6	73.1	32.0	78.4	62.8

(b)

	Axe	Bachata	Bolero	Forro	Gaucha	Merengue	Pagode	Salsa	Sertaneja	Tango	Pr
Axe	92.0	0.0	2.0	2.0	4.0	2.0	12.0	2.0	8.0	0.0	74.2
Bachata	0.0	100.0	2.0	0.0	2.0	0.0	0.0	6.0	0.0	0.0	90.9
Bolero	2.0	0.0	76.0	2.0	4.0	2.0	0.0	8.0	20.0	8.0	62.3
Forro	0.0	0.0	0.0	86.0	6.0	0.0	2.0	0.0	4.0	0.0	87.8
Gaucha	0.0	0.0	0.0	0.0	68.0	4.0	0.0	0.0	0.0	0.0	94.4
Merengue	0.0	0.0	0.0	0.0	2.0	88.0	0.0	4.0	0.0	0.0	93.6
Pagode	0.0	0.0	4.0	2.0	0.0	0.0	78.0	0.0	2.0	0.0	90.7
Salsa	2.0	0.0	10.0	2.0	2.0	4.0	2.0	80.0	0.0	0.0	78.4
Sertaneja	4.0	0.0	6.0	6.0	12.0	0.0	4.0	0.0	64.0	0.0	66.7
Tango	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	2.0	92.0	95.8
F	82.1	95.2	68.5	86.9	79.1	90.7	83.9	79.2	65.3	93.9	82.4

(c)

	Axe	Bachata	Bolero	Forro	Gaucha	Merengue	Pagode	Salsa	Sertaneja	Tango	Pr
Axe	36.4	0.0	0.0	3.5	3.2	1.6	0.0	1.6	1.4	0.0	77.4
Bachata	1.5	85.5	1.6	0.0	1.6	1.6	0.0	0.0	5.8	0.0	86.9
Bolero	16.7	6.5	77.8	10.5	4.8	1.6	13.1	0.0	27.5	45.7	35.5
Forro	0.0	1.6	4.8	52.6	6.3	0.0	6.6	0.0	7.2	0.0	63.8
Gaucha	13.6	1.6	1.6	1.8	69.8	1.6	4.9	0.0	15.9	0.0	62.0
Merengue	0.0	1.6	0.0	1.8	0.0	88.7	4.9	1.6	0.0	0.0	90.2
Pagode	1.5	1.6	1.6	8.8	6.3	0.0	52.5	3.2	5.8	0.0	64.0
Salsa	21.2	1.6	3.2	1.8	4.8	4.8	13.1	93.5	4.3	3.7	60.4
Sertaneja	9.1	0.0	9.5	19.3	3.2	0.0	4.9	0.0	31.9	0.0	44.0
Tango	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	50.6	100.0
F	49.5	86.2	48.8	57.7	65.7	89.4	57.7	73.4	37.0	67.2	63.2

(d)

Fig. 3. FoM for deep learning systems with two different partitioning strategies of *LMD*. Interpretation as in Fig. 2, but note that in this case we are using the deep learning systems as the classifiers, instead of performing classification using a random forest with features derived from hidden layer activations. (a) DNN random partitioning. (b) DNN artist-filtered partitioning. (c) CDNN random partitioning. (d) CDNN artist-filtered partitioning.

with the parameter

$$\epsilon(\text{SNR}) = \frac{\frac{1}{N}\sqrt{\sum_{n=0}^{N-1} \|X_n\|_2^2}}{10^{\text{SNR}/20}} \tag{8}$$

limiting the maximum acceptable perturbation caused by the adversary. The loss function in (6) is the cross-entropy loss function, $\mathcal{L}(X_n, y|\Theta) := -\log P_n(y|X_n, \Theta)$, which we use in training our (C)DNNs. Given the network parameters Θ, this adversary can compute the derivative of this loss function by back-propagating derivatives through the network. This suggests that our adversary can accomplish its goal by searching for a new input sequence $\hat{X}$ via gradient descent on the loss function with any label y. This is the approach used by Szegedy *et al.* [10] in the context of image object recognition.

A local minimum of (6) can be found using projected gradient descent, initialised with $\hat{X}^{(0)} \leftarrow X$, and iterating

$$\hat{X}^{(k+1)} \leftarrow \mathcal{P}_{\mathcal{C}}(\hat{X}^{(k)} + \mu \nabla \mathcal{L}(\hat{X}^{(k)}, y|\Theta)) \tag{9}$$

where the scalar μ is the gradient descent step size, and $\mathcal{P}_{\mathcal{C}}(\cdot)$ computes the least squares projection of its argument onto the set $\mathcal{C}(X)$ defined in (7). Note that we define operations on sequences element-wise, e.g., $\nabla \mathcal{L}(\hat{X}^{(k)}, y|\Theta) = (\nabla \mathcal{L}(\hat{X}_n^{(k)}, y|\Theta))_{n=0}^{N-1}$.

Algorithm 1 From exemplar sequence X search for valid adversarial sequence $\hat{X}$ with maximal perturbation SNR in at most $k_{\max}$ steps that makes a (C)DNN with parameters Θ apply label y with confidence $R_{\min}$.

```
1: parameters: $y$, SNR, $\mu$, $R_{\min}$, $\Theta$, $k_{\max}$
2: init: $\hat{X}^{(0)} = X, k = 0$
3: repeat
4:     $V \leftarrow \hat{X}^{(k)} + \mu\nabla\mathcal{L}(\hat{X}^{(k)}, y|\Theta)$ {Gradient step}
5:     $W \leftarrow \mathcal{P}_{GL}(\max(0, V))$ {Find valid sequence}
6:     $\nu \leftarrow \max(0, \frac{1}{N}\sqrt{\sum_{n=0}^{N-1} ||W_n - \hat{X}_n||_2^2 / \epsilon(\text{SNR})} - 1)$
       {Lagrange mult.}
7:     $\hat{X}^{(k+1)} \leftarrow (1+\nu)^{-1}(W + \nu X)$ {SNR constraint}
8:     $k \leftarrow k + 1$
9: until $\frac{1}{N}\sum_{n=0}^{N-1} P_n(y|\hat{X}_n^{(k+1),\Theta}) \geq R_{\min}$ or $k = k_{\max}$
10: return: $\hat{X}^{(k)}$
```

The main difficulty with this approach is that not all sequences $\hat{X}$ can be mapped back to valid time-domain signals $\hat{x}$. This is because the analysis in (1) uses overlapping windows, which causes adjacent elements in the sequence X to become dependent. This means that individual elements from the sequence X cannot be adjusted arbitrarily if we want X to have an analog in the time-domain. Therefore, in order to generate valid adversarial examples, we include an additional processing step that projects the sequence $\hat{X}$ onto the space of time-frequency coefficients arising from valid time-domain sequences. This is done using the Griffin and Lim algorithm [61], which seeks to minimize

$$\mathcal{P}_{GL}(\hat{X}) = \min_{Z \in \mathcal{X}} \sum_{n=0}^{N-1} ||Z_n - \hat{X}_n||_2^2 \tag{10}$$

where $\mathcal{X} = \{X = (X_n)_{n=0}^{N-1} : \mathcal{P}_{GL}(X) = X\}$ denotes the set of all valid sequences. This minimization can be performed using alternating projections, and we have found that in practice it is sufficient to apply a single set of projections. We do this by first rebuilding a complex valued time-frequency representation from the sequence $\hat{X}$

$$U[m, u] = \begin{cases} \hat{X}_{u/T}[m, u \bmod T]e^{j\Phi[m,u]}, & 0 \leq m < D \\ \hat{X}_{u/T}[D - m, u \bmod T]e^{j\Phi[m,u]}, & D \leq m < L. \end{cases} \tag{11}$$

where $D = L/2 + 1$ and $\Phi[m, u] \triangleq \angle\mathcal{F}(x)$ is the phase from the exemplar's Fourier transform. The inverse Fourier transform $\mathcal{F}^{-1}(U)$ is a time-domain signal, and so the Fourier transform of this signal, $\mathcal{F} \circ \mathcal{F}^{-1}(U)$, will yield a valid DFT spectrum that can be used to build a valid input sequence for our (C)DNN, i.e., by replacing $\mathcal{F}(x)$ by $\mathcal{F} \circ \mathcal{F}^{-1}(U)$ in (2).

The pseudo-code in Alg. 1 summarizes our adversary. The algorithm may be terminated when the mean posterior of the target adversarial label exceeds the threshold $R_{\min}$, or after a maximum number of epochs $k_{\max}$ (in which case an adversarial example cannot be found above the minimum SNR).

B. *Training With Adversaries for Music Audio*

As per [10] and [15], we can attempt to use our adversary as a regularizer to create systems robust against adversarial inputs. In particular, an adversary is used to generate a (possibly) infinite supply of new samples during training (which the system should learn to become robust to). The iterative procedure for generating adversaries in Alg 1 is too slow to be practical for training; therefore, we apply the single gradient step procedure suggested in [15]. In our experience, this procedure often generates inputs that confuse the network, although not typically with a high confidence. The pseudo-code in Alg. 2 illustrates our training algorithm, where $(\mathbf{X}, \mathbf{Y})$ represent the training data, i.e., the set of input audio sequences and their labels, and $\hat{\mathbf{Y}}$ is a set of adversarial labels.

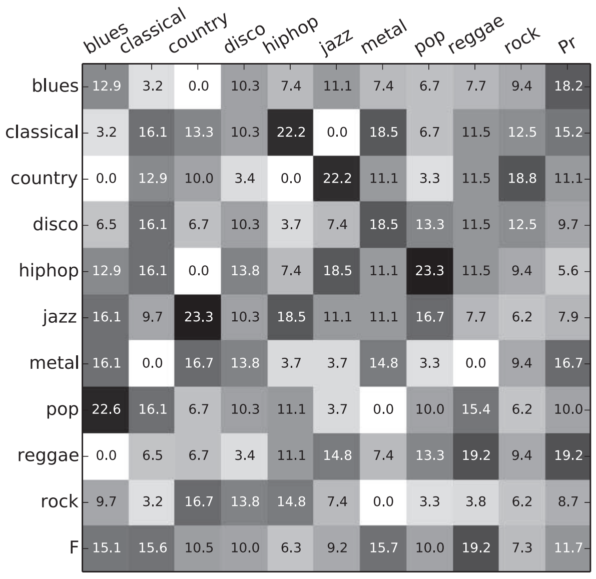
Algorithm 2 Train (C)DNN using database of labeled sequences $(\mathbf{X}, \mathbf{Y})$ and fast adversarial generation [15], with ε and μ the gradient descent step sizes for adjusting the adversarial inputs and network weights, respectively.

```
1: parameters: $\varepsilon$, $\mu$
2: init: (C)DNN parameters $\Theta$ to small random weights
3: repeat
4:    select $\hat{\mathbf{Y}}$ uniformly $\{1, \ldots, K\}^N$
5:    $\hat{\mathbf{X}} \leftarrow \mathbf{X} + \varepsilon\nabla\mathcal{L}(\mathbf{X}, \hat{\mathbf{Y}}|\Theta)$ {Generate adversarial ex.}
6:    $\Theta \leftarrow \Theta + \mu\nabla\mathcal{L}(\hat{\mathbf{X}}, \mathbf{Y}|\Theta)$ {Model update}
7: until Stopping condition, e.g., max no. of epochs
```

IV. Experimental Results

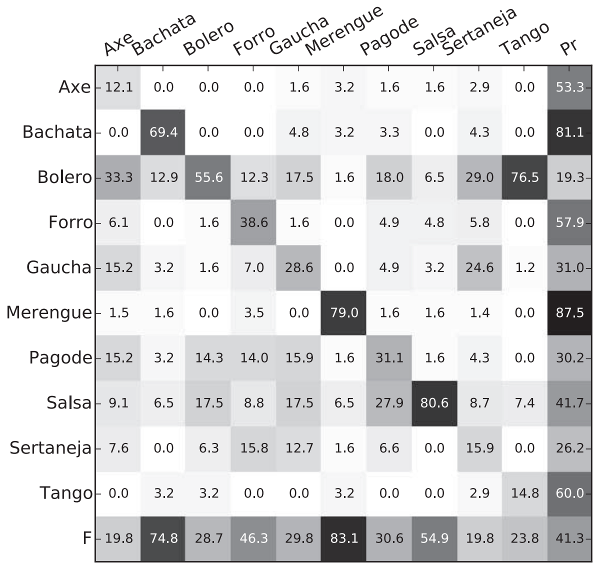
We can design an adversary (Alg. 1) such that it will attempt to make a system behave in different ways. For instance, an adversary could attempt to perturb an input within some limit (SNR) such that the (C)DNN makes a high-confidence classification ($R_{\min} \approx 1$) that is correct with probability p. Another adversary could attempt to make the system label any input using the same label. We can also make an ensemble of adversaries such that they produce adversarial examples that a (C)DNN classifies in every possible way.

We define our adversaries using: $R_{\min} = 0.9$, SNR $= 15$ dB, $\mu = 0.1$, and $k_{\max} = 100$, and with the directive to make the (C)DNN correct with probability $p = 0.1$. More concretely, for each test observations, the adversary draws uniformly one of the dataset labels y, then seeks to find in no more than $k_{\max} = 100$ iterations using step size $\mu = 0.1$ a valid perturbation no larger than 15 dB SNR, and which the (C)DNN labels as y with confidence $R_{\min} = 0.9$. Fig. 4(a) shows the FoM of the DNN-based classification system in Fig. 2(b), but with input intercepted by this adversary. Note that in this case the classification is performed by a random forest classifier using the aggregated hidden layer activations, but the adversary is unaware of this. In other words, it is only trying to force the DNN to misclassify individually perturbed inputs. Compared with the mean recall of 0.49 in Fig. 2(b), we see our adversary has successfully confused the random forest classifier to be no better than random. Fig. 5 shows one of the adversarial examples from this experiment. Apart from some significant high-frequency deviations, the spectrum of the adversary is very similar to that of the original.



	blues	classical	country	disco	hiphop	jazz	metal	pop	reggae	rock	Pr
blues	12.9	3.2	0.0	10.3	7.4	11.1	7.4	6.7	7.7	9.4	18.2
classical	3.2	16.1	13.3	10.3	22.2	0.0	18.5	6.7	11.5	12.5	15.2
country	0.0	12.9	10.0	3.4	0.0	22.2	11.1	3.3	11.5	18.8	11.1
disco	6.5	16.1	6.7	10.3	3.7	7.4	18.5	13.3	11.5	12.5	9.7
hiphop	12.9	16.1	0.0	13.8	7.4	18.5	11.1	23.3	11.5	9.4	5.6
jazz	16.1	9.7	23.3	10.3	18.5	11.1	11.1	16.7	7.7	6.2	7.9
metal	16.1	0.0	16.7	13.8	3.7	3.7	14.8	3.3	0.0	9.4	16.7
pop	22.6	16.1	6.7	10.3	11.1	3.7	0.0	10.0	15.4	6.2	10.0
reggae	0.0	6.5	6.7	3.4	11.1	14.8	7.4	13.3	19.2	9.4	19.2
rock	9.7	3.2	16.7	13.8	14.8	7.4	0.0	3.3	3.8	6.2	8.7
F	15.1	15.6	10.5	10.0	6.3	9.2	15.7	10.0	19.2	7.3	11.7

(a)



	Axe	Bachata	Bolero	Forro	Gaucha	Merengue	Pagode	Salsa	Sertaneja	Tango	Pr
Axe	12.1	0.0	0.0	0.0	1.6	3.2	1.6	1.6	2.9	0.0	53.3
Bachata	0.0	69.4	0.0	0.0	4.8	3.2	3.3	0.0	4.3	0.0	81.1
Bolero	33.3	12.9	55.6	12.3	17.5	1.6	18.0	6.5	29.0	76.5	19.3
Forro	6.1	0.0	1.6	38.6	1.6	0.0	4.9	4.8	5.8	0.0	57.9
Gaucha	15.2	3.2	1.6	7.0	28.6	0.0	4.9	3.2	24.6	1.2	31.0
Merengue	1.5	1.6	0.0	3.5	0.0	79.0	1.6	1.6	1.4	0.0	87.5
Pagode	15.2	3.2	14.3	14.0	15.9	1.6	31.1	1.6	4.3	0.0	30.2
Salsa	9.1	6.5	17.5	8.8	17.5	6.5	27.9	80.6	8.7	7.4	41.7
Sertaneja	7.6	0.0	6.3	15.8	12.7	1.6	6.6	0.0	15.9	0.0	26.2
Tango	0.0	3.2	3.2	0.0	0.0	3.2	0.0	0.0	2.9	14.8	60.0
F	19.8	74.8	28.7	46.3	29.8	83.1	30.6	54.9	19.8	23.8	41.3

(b)

Fig. 4. For the DNN-based classifier in Fig. 2(b) and the CDNN in Fig. 3(d), but with all input intercepted by an adversary intent on making the maximum posterior classification correct with probability $p = 0.1$. For this adversary, $R_{\min} = 0.9$, SNR $= 15$ dB, $\mu = 0.1$, and $k_{\max} = 100$. Sub-captions show the resulting SNR (mean $\pm$ standard deviation). (a) *GTZAN* fault-filtered: 23.0 $\pm$ 4.5 dB. (b) *LMD* artist-filtered: 15.78 $\pm$ 4.65 dB.

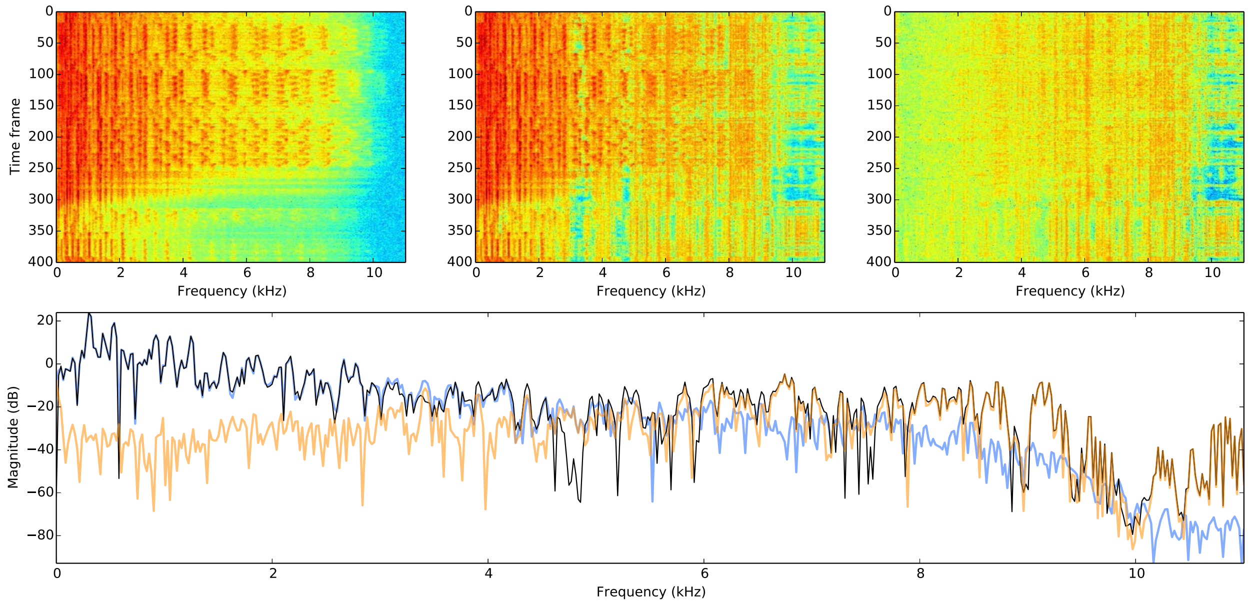


Fig. 5. Top left: spectrogram excerpt from *GTZAN* Classical "21" (Mozart, Symphony No. 39 Finale) that the DNN-based system in Fig. 2(b) classifies as *Classical*. Top middle: spectrogram of adversarial example classified as *Reggae*. Top right: spectrogram of the difference of the two. Bottom: magnitude spectrum of one frame (1024 samples) of the original (light blue), adversarial example (black), and difference (orange). Note that all excerpts in *GTZAN* have a sampling rate of 22050 Hz. The SNR $= 21.1$ dB.

Fig. 4(b) shows the FoM of the CDNN classification system in Fig. 3(d) attacked by the same adversary. In this case, the CDNN proved more difficult to fool, but still the adversary is able to significantly reduce the mean recall from 0.63 down to 0.41 with high confidence classifications at rather high SNR. If we reduce the minimum confidence $R_{\min} = 0.5$ and lessen the SNR constraint to -300 dB, then the adversary makes the CDNN perform even worse: a mean recall of 0.28 with a mean SNR of 11.15 $\pm$ 8.32 dB.

For the same system in Fig. 2(b), and using $R_{\min} = 0.9$, $SNR = 15$ dB, $\mu = 0.1$ and $k_{\max} = 100$, we show in [14] that we able to create adversaries that make the system always right, always wrong, and always select "Jazz." Table II shows the results of two ensembles of adversaries, each intent on making the system in Fig. 2(b) choose one of every label in *GTZAN* for the same music with $SNR = 15$ dB, $\mu = 0.1$ and $k_{\max} = 100$. The adversaries of one ensemble insist upon a classification confidence of at least $R_{\min} = 0.5$; and in the other of at least 0.9. These music recordings are the same 30-second excerpts used in [11]. We see that in all case but one, the ensembles are able to elicit high confidence classifications from the system with minor perturbations of the input. We also see that larger per-

TABLE II
SNR OF PERTURBATIONS PRODUCED BY TWO ENSEMBLES OF ADVERSARIES THAT INTERCEPT THE INPUT TO THE SYSTEM IN FIG. 2(B) AND HAVE IT PRODUCE ALL POSSIBLE LABELS WITH CONFIDENCE THRESHOLDS $R_{\min} = 0.5$ ($R_{\min} = 0.9$ IN BRACKETS). THE AVERAGE SNR IS 34.5 (26.8 DB). THIS TABLE CAN BE HEARD AT HTTP://WWW.EECS.QMUL.AC.UK/~STURM/RESEARCH/DNN_ADVERSARIES

	Classification in *GTZAN*									
Music excerpt	*Blues*	*Classical*	*Country*	*Disco*	*Hiphop*	*Jazz*	*Metal*	*Pop*	*Reggae*	*Rock*
Little Richard, "Last Year's Race Horse"	32 (23)	29 (23)	36 (25)	36 (26)	36 (25)	33 (24)	32 (24)	31 (25)	42 (26)	36 (25)
Rossini, "William Tell Overture"	32 (25)	37 (30)	40 (29)	43 (28)	34 (24)	36 (29)	33 (25)	34 (26)	37 (26)	37 (28)
Willie Nelson, "A Horse Called Music"	25 ( )	25 (20)	30 (27)	30 (20)	26 (19)	30 (25)	27 (23)	21 (20)	30 (23)	29 (23)
Simian Mobile Disco, "10000 Horses Can't Be Wrong"	31 (30)	36 (31)	38 (32)	45 (34)	41 (33)	40 (32)	33 (31)	47 (34)	42 (33)	38 (33)
Rubber Bandits, "Horse Outside"	27 (27)	27 (27)	36 (29)	42 (31)	38 (29)	34 (28)	32 (28)	37 (29)	36 (29)	35 (29)
Leonard Gaskin, "Riders in the Sky"	32 (23)	30 (25)	32 (23)	35 (25)	31 (22)	35 (29)	34 (23)	26 (23)	35 (25)	35 (24)
Jethro Tull, "Heavy Horses"	29 (26)	28 (26)	40 (29)	42 (29)	38 (28)	36 (28)	34 (28)	34 (28)	37 (28)	36 (29)
Echo and The Bunnymen, "Bring on the Dancing Horses"	29 (25)	28 (26)	38 (28)	43 (28)	35 (26)	34 (26)	33 (26)	33 (26)	36 (27)	38 (28)
Count Prince Miller, "Mule Train"	32 (30)	29 (30)	41 (33)	37 (34)	43 (33)	36 (31)	33 (31)	42 (34)	40 (33)	33 (33)
Rolling Stones, "Wild Horses"	30 (22)	32 (24)	37 (25)	40 (25)	31 (22)	34 (25)	31 (26)	32 (23)	37 (25)	37 (26)

TABLE III
RESULTS OF APPLYING ADVERSARY TO MAKE SYSTEMS IN FIG. 3(B) AND (D) ALWAYS INCORRECT, AND AFTER TRAINING WITH ADVERSARY (ALG. 2)

Deep Learning System	*Mean Recall*	*Mean Recall w/ Adversary*	*SNR (dB) mean ± std. dev.*
DNN-LMD Fig. 3(b)	0.63	0.03	37.8±4.6
DNN-LMD+ADV	0.55	0.06	36.5±5.4
CDNN-LMD Fig. 3(d)	0.63	0.21	9.62±5.8
CDNN-LMD+ADV	0.56	0.21	9.74±6.4

turbations are required on average when the adversaries insist on a higher minimum confidence: 34.5 dB for a confidence of at least $R_{\min} = 0.5$, and 26.8 dB for a confidence of at least $R_{\min} = 0.9$.

These results can be heard here: http://www.eecs.qmul.ac.uk/~sturm/research/DNN_adversaries. We find that the perturbations caused by these adversaries are certainly perceptible, unlike those found for image data in [10] and [15]; however, the distortion can be very minor, and the music remains *exactly* the same, e.g., pitches, rhythm, lyrics, instrumentation, dynamics, and style all remain the same.

We now perform an experiment to compare (C)DNNs trained with adversarial examples (Alg. 2) to the systems in Fig. 3(b), (d). To do this we test the response of these systems against an adversary aimed at *always* eliciting an incorrect response. (This is different from the adversaries used above, which seek to make a network correct with probability $p = 0.1$.) For this experiment, we set $R_{\min} = 0.5$ and SNR to -300 dB in order to allow arbitrarily large perturbations to force misclassifications. Table III illustrates the results of this experiment from which we observe several interesting results. Column 2 shows the mean recall on the original test set (with no adversary present). We see that training against adversarial examples leads to a slight deflation in accuracy on the test data. Column 3 shows the mean recall of these systems against an adversary intent on forcing a 100% error rate. We see that the CDNN systems are more robust to this adversary, and that the systems trained against adversarial examples confer little to no advantage. Column 4 shows the average perturbation size of the adversarial examples that led to misclassifications. We notice that larger perturbations (corresponding to lower SNRs) were required to make the CDNN systems misclassify inputs. The minimum SNR produced was 0.11 dB, while the maximum was 47.6 dB. The results of this experiment point to the conclusions that a) the CDNN systems are more robust to this adversary; and b) training against adversarial examples (contrary to what we hypothesized) does not seem reduce the misclassification rate against new adversarial examples. The latter result could be due to the fact that the adversarial examples generated using Alg. 2 tend to be lower confidence (and hence are weaker adversaries) than those generated using Alg. 1.

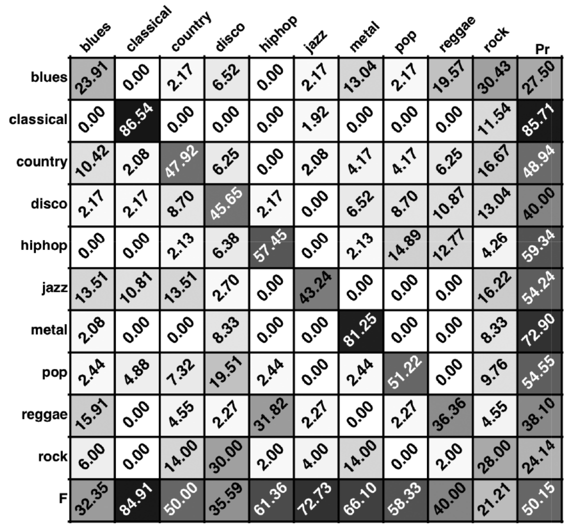
V. DISCUSSION

Returning to the broadest question motivating our work, we seek to measure the contribution of deep learning to music content analysis. The previous sections describe a series of experiments we have conducted using deep learning systems of a variety of architectures, which we have trained and tested in two different partitions of two benchmark music datasets. We have evaluated the robustness of these systems to adversaries that have complete knowledge of the classifiers, and have also investigated the use of an adversary in the training of deep learning systems.

A. Fault-Filtering and Hand-Crafted Features

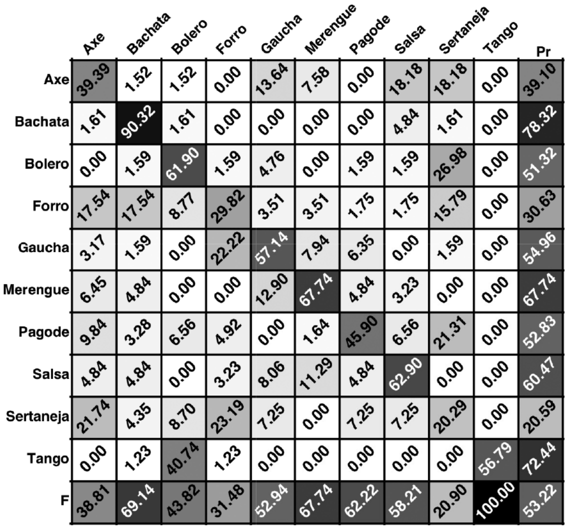
Our experimental results in Fig. 2 and Table I are essentially reproductions of those reported in [7]. Based on the results of their experiments with random partitionings of *GTZAN*, Sigtia *et al.* [7] claim that their DNN-based systems learn features that "better represent the audio" than standard or "hand-crafted" features, e.g., those referenced in [62] like MFCCs. Similar conclusions are made about the deep learning systems in [2], also based on experiments using a random partitioning of *GTZAN*. However, we see in Fig. 2 and Table I that when we consider the faults in the *GTZAN* dataset and partition it along artist lines, as for the *LMD* dataset in Fig. 3, our deep learning systems perform significantly worse. This is an expected outcome [12], [34], [36], but the artist information in *GTZAN* was not available until 2012 [63].

This motivates the question of whether DNN-based systems really do perform better than that of a classifier using standard, low-level and "hand-crafted" features. To examine this, we build baseline systems that use low-level features, and train and test them in the same fault-filtered partition of *GTZAN* as in Fig. 2(b), and the artist-filtered partition of *LMD* as in [Fig. 3(b),(d)]. Mimicking [2], [7], we compute these features based on a short-time analysis using 46 ms frames hopped by 50%. From each frame we extract the first 13 Mel-frequency cepstral coefficients (MFCCs) and zero-crossings, and compute their mean and variance over five-second texture windows



	blues	classical	country	disco	hiphop	jazz	metal	pop	reggae	rock	Pr
blues	23.91	0.00	2.17	6.52	0.00	2.17	13.04	2.17	19.57	30.43	27.50
classical	0.00	86.54	0.00	0.00	0.00	1.92	0.00	0.00	0.00	11.54	85.71
country	10.42	2.08	47.92	6.25	0.00	2.08	4.17	4.17	6.25	16.67	48.94
disco	2.17	2.17	8.70	45.65	2.17	0.00	6.52	8.70	10.87	13.04	40.00
hiphop	0.00	0.00	2.13	6.38	57.45	0.00	2.13	14.89	12.77	4.26	59.34
jazz	13.51	10.81	13.51	2.70	0.00	43.24	0.00	0.00	0.00	16.22	54.24
metal	2.08	0.00	0.00	8.33	0.00	0.00	81.25	0.00	0.00	8.33	72.90
pop	2.44	4.88	7.32	19.51	2.44	0.00	2.44	51.22	0.00	9.76	54.55
reggae	15.91	0.00	4.55	2.27	31.82	2.27	0.00	2.27	36.36	4.55	38.10
rock	6.00	0.00	14.00	30.00	2.00	4.00	14.00	0.00	2.00	28.00	24.14
F	32.35	84.91	50.00	35.59	61.36	72.73	66.10	58.33	40.00	21.21	50.15

(a)



	Axe	Bachata	Bolero	Forro	Gaucha	Merengue	Pagode	Salsa	Sertaneja	Tango	Pr
Axe	39.39	1.52	1.52	0.00	13.64	7.58	0.00	18.18	18.18	0.00	39.10
Bachata	1.61	90.32	1.61	0.00	0.00	0.00	0.00	4.84	1.61	0.00	78.32
Bolero	0.00	1.59	61.90	1.59	4.76	0.00	1.59	1.59	26.98	0.00	51.32
Forro	17.54	17.54	8.77	29.82	3.51	3.51	1.75	1.75	15.79	0.00	30.63
Gaucha	3.17	1.59	0.00	22.22	57.14	7.94	6.35	0.00	1.59	0.00	54.96
Merengue	6.45	4.84	0.00	0.00	12.90	67.74	4.84	3.23	0.00	0.00	67.74
Pagode	9.84	3.28	6.56	4.92	0.00	1.64	45.90	6.56	21.31	0.00	52.83
Salsa	4.84	4.84	0.00	3.23	8.06	11.29	4.84	62.90	0.00	0.00	60.47
Sertaneja	21.74	4.35	8.70	23.19	7.25	0.00	7.25	7.25	20.29	0.00	20.59
Tango	0.00	1.23	40.74	1.23	0.00	0.00	0.00	0.00	0.00	56.79	72.44
F	38.81	69.14	43.82	31.48	52.94	67.74	62.22	58.21	20.90	100.00	53.22

(b)

Fig. 6. As in Fig. 2, FoM for majority vote of minimum Mahalanobis distance classification of mean and variances over 5-second "texture" windows of zero-crossings and the first 13 MFCCs computed from 46 ms windows hopped 50%. (a) *GTZAN* fault-filtered. (b) *LMD* artist-filtered.

(which are also hopped by 50%). We combine the features of the training and validation sets of the fault-filtered partition of *GTZAN*, and the artist filtered partition of *LMD*. Both systems use a minimum Mahalanobis distance classifier, and assign a class by majority vote from the classifications of the individual texture windows. Fig. 6 shows the FoM produced by these baseline systems. We see that for *GTZAN* it actually reproduces more ground truth than the DNN in Fig. 2(b) and all but one in Table I. Our simple baseline system for *LMD* reproduces much less ground truth than the (C)DNN in (Fig. 3(b), (d)). Nonetheless, we have no reason to accept the conclusion that deep learning features "perform better" than "hand-crafted" features for the particular architectures considered here and those in [2], [7]. Different experiments are needed to address such a conclusion.

B. Explaining Figures of Merit

A tempting conclusion is that since the normalised classification accuracies in Figs. 2(b) and 3(d) are extremely unlikely to arise by chance ($p< 10^{-62}$ for *GTZAN* and $p< 10^{-290}$ for *LMD* by a Binomial test) it is therefore entirely reasonable to reject the hypothesis that our (C)DNN are choosing outputs at random. Hence, one might argue that these (C)DNN must have learned features that are "relevant" to music genre recognition [2], [4], [7]. This argument appears throughout the MIR research discipline [12], and turns on the strong assumption that there are only two ways a system can reproduce the ground truth of a dataset: by chance or by learning to solve a specific problem thought to be well-posed by a cleanly labeled dataset [39]. In fact, there is a third way a system can reproduce the ground truth of a music dataset: by learning to exploit characteristics shared between the training and testing datasets that arise not from a relationship in the real world, but from the curation and partitioning of a dataset in the experimental design of an evaluation [11], [12], [40]. Since the evaluations producing Figs. 2 and 3, as well as all results in [2], [7], not to mention a significant number of published studies in MIR [12], do not control for this third way, we cannot validly conclude upon the "relevance" of whatever has been learned by these music content analysis systems.

A notion of this problem is given by the significant decreases in the FoM we measure when partitioning *GTZAN* and *LMD* along artist lines. By doing so, we are controlling for *some* independent variables that a system might be exploiting to reproduce ground truth, but which arguably have little relevance to the high-level labels of the dataset [12]. More concretely, consider that all 100 excerpts labeled *Pop* in *GTZAN* come from recordings of music by four artists, 25 from each artist. If we train and test a system on a random partition of *GTZAN*, we cannot know whether the system is recognising *Pop*, recognising the artist, or recognising other aspects that may or may not be related to *Pop*. If we train a system instead with *Pop* excerpts by three artists, test with the *Pop* excerpts by the fourth artist, then we might be testing something closer to *Pop* recognition. This all depends on defining what knowledge is relevant to the problem.

A common retort to these arguments is that a system should be able to reproduce ground truth "by any means." One thereby defines "relevant knowledge" as *any* correlations that helps a system reproduce an amount of dataset ground truth that is inconsistent with chance. However, this can lead to circular reasoning: system X has learned "relevant knowledge" because it reproduces Y amount of ground truth; system X reproduces Y amount of ground truth because it has learned "relevant knowledge." It is also deaf to one of the major aims of research in music content analysis [17]: "to make music, or information about music, easier to find." If a music content analysis system is describing music in ways that do not align with those of its users, then its usability is in jeopardy no matter its FoM in benchmark datasets [38], [64]. Finally, this means that the problem thought to be well-posed by a cleanly labeled dataset can be many things simultaneously—which leads to the problem of how to validly compare apples and oranges [39]. In other words, why compare systems when they are *solving* different problems? This also applies to the comparisons above with the FoM in Fig. 6.

While we currently do not know whether our (C)DNN systems in Fig. 3 are exploiting "irrelevant" characteristics in *LMD*, our experimental results with adversaries in Fig. 4 and 5, and Tables II and III, indicate that their decision machinery is incredibly sensitive in very strange ways. Our adversaries are able to fool the high-performing deep learning systems by perturbing their input in minor ways. Auditioning the results in Table II show that while the music in each recording remains exactly the same, and the perturbations are very small, the DNN is nearly always fooled into choosing with high confidence every class it has supposedly learned to identify. The CDNN is similarly defeated by our adversary; however, it is quite notable that it requires perturbations of far lower SNR than does the DNN. We are currently studying the reasons for this.

Our application of adversaries here is close to the "method of irrelevant transformations" that we apply in [11], [13], [40] to assess the internal models of music content analysis systems, and to test the hypothesis, "the system is using relevant criteria to make its decisions." In [11], we take a brute force approach whereby we apply random but linear time-invariant and minor filtering to inputs of systems trained in three different music recording datasets until their FoM becomes perfect or random. We also make each system apply every one of its classes to the same music recordings in Table II.[4] In [40], we instead apply subtle pitch-preserving time-stretching of music recordings to fool a deep learning system trained in the benchmark music dataset *BALLROOM* [65]. We find that through such a transformation we can make the system perform perfectly or no better than random by applying tempo changes of at most 6% to test dataset recordings. We find a similar result for the same kind of deep learning system but trained in *LMD* [13].

C. On Potemkin Villages and Horses

Our adversary in Alg. 1 moves right to the achilles heel of a deep learning system, coaxing it to behave in arbitrary ways for an input simply by making minor perturbations to the sampled audio waveform that have no effect on the music content it embodies. We observe in Fig. 5 and auditioning Table II that the low- to mid-frequency content of adversarial examples differs very little from the original recordings, but find more significant differences in the high-frequency spectra. This suggests that the distribution of energy in the high-frequency spectrum has significant impact on the decision machinery of our (C)DNN. The apparent relevance of such slight characteristics in proportion to that of the actual musical content of a music recording does not bode well for one of the most important aims of machine learning: *generalization*.

As observed by Goodfellow *et al.* [15] in their deep learning systems taught to recognize objects in images, the impressive FoM we measure of our deep learning systems may be merely a colourful "Potemkin village." Employing an adversary to scratch a little below the surface reveals the FoM to be curiously hollow. A system that appears to be solving a complex problem but actually is not is what we term a "horse" [11], which is a nod to the famous horse Clever Hans: a real horse that *appeared* to be a capable mathematician but was merely responding to involuntary cues that went undetected because his public demonstrations had no validity to attest to such an ability. Measuring the number of correct answers Hans gives in an uncontrolled environment does not give reason to conclude he comprehends what he appears to be doing. It is the same with the experiments we perform above with systems labelling observations in *GTZAN* and *LMD*. In fact, Goodfellow *et al.* [15] come to the same conclusion: "The existence of adversarial examples suggests that … being able to correctly label the test data does not imply that our models truly understand the tasks we have asked them to perform" [15]. This observation deserves to be repeated.

[4] These results can be auditioned here: http://www.eecs.qmul.ac.uk/~sturm/research/TM_expt2/index.html

VI. Conclusion

In this article, we have shown how to adapt the adversary of Szegedy *et al.* [10], [15] to work within the context of music content analysis using deep learning. We have shown how our adversary is effective at fooling specific deep learning systems of different architectures, trained on different benchmark datasets. We find our convolutional networks are more robust against this adversary than our deep neural networks. We have also sought to employ the adversary as part of the training of these systems, but find the resulting systems remain sensitive to our adversaries.

It is of course not very popular for one to be an "adversary" to research, moving quickly to refute conclusions and break systems reported in the literature; however, we insist that breaking systems ultimately leads to progress. Considerable insight can be gained by looking behind the veil of performance metrics in an attempt to determine the mechanisms by which a system operates, and whether the evaluation is any valid reflection of the qualities we wish to measure. Such probing is necessary if we are truly interested in ascertaining what a system has learned to do, what its vulnerabilities might be, how it compares to competing systems supposedly solving the same problem, and how well we can expect it to perform when used in real-world applications.

Acknowledgment

This publication only reflects the authors' views.

References

[1] H. Lee, Y. Largman, P. Pham, and A. Y. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Proc. NIPS*, Dec. 2009, pp. 1096–1104.
[2] P. Hamel and D. Eck, "Learning features from music audio with deep belief networks," in *Proc. ISMIR*, 2010, pp. 339–344.
[3] X. Yang, Q. Chen, S. Zhou, and X. Wang, "Deep belief networks for automatic music genre classification," in *Proc. Interspeech*, 2011, pp. 2433–2436.
[4] E. J. Humphrey, J. P. Bello, and Y. LeCun, "Feature learning and deep architectures: New directions for music informatics," *J. Intell. Inf. Syst.*, vol. 41, no. 3, pp. 461–481, 2013.
[5] A. van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Proc. NIPS*, 2013, pp. 2643–2651.
[6] A. Pikrakis, "A deep learning approach to rhythm modeling with applications," in *Proc. Int. Workshop Mach. Learn. Music*, 2013, pp. 1–4.
[7] S. Sigtia and S. Dixon, "Improved music feature learning with deep neural networks," in *Proc. ICASSP*, May 2014, pp. 6959–6963.
[8] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," in *Proc. ICASSP*, May 2014, pp. 6964–6968.
[9] C. Zhang, G. Evangelopoulos, S. Voinea, L. Rosasco, and T. Poggio, "A deep representation for invariance and music classification," in *Proc. ICASSP*, May 2014, pp. 6984–6988.

[10] C. Szegedy *et al.*, "Intriguing properties of neural networks," in *Proc. ICLR*, 2014, pp. 1–9.
[11] B. L. Sturm, "Making explicit the formalism underlying evaluation in music information retrieval research: A look at the {MIREX} automatic mood classification task," in *Proc. Comput. Music Modeling Res.*, 2014, pp. 89–104.
[12] B. L. Sturm, "The state of the art ten years after a state of the art: Future research in music information retrieval," *J. New Music Res.*, vol. 43, no. 2, pp. 147–172, 2014.
[13] B. L. Sturm, C. Kereliuk, and J. Larsen, "¿El Caballo Viejo? Latin genre recognition with deep learning and spectral periodicity," in *Proc. Int. Conf. Math. Comput. Music*, 2015, pp. 335–346.
[14] C. Kereliuk, B. L. Sturm, and J. Larsen, "Deep learning, audio adversaries, and music content analysis," in *Proc. WASPAA*, 2015, pp. 1–10.
[15] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. ICLR*, 2015, pp. 1–11.
[16] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," *CoRR*, Dec. 2014 [Online]. Available: http://arxiv.org/abs/1412.5068
[17] M. Casey *et al.*, "Content-based music information retrieval: Current directions and future challenges," *Proc. IEEE*, vol. 96, no. 4, pp. 668–696, Apr. 2008.
[18] A. Wang, "An industrial strength audio search algorithm," in *Proc. Int. Soc. Music Info. Retrieval*, Oct. 2003, pp. 1–7.
[19] M. Casey, C. Rhodes, and M. Slaney, "Analysis of minimum distances in high-dimensional musical spaces," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 16, no. 5, pp. 1015–1028, Jul. 2008.
[20] S. Ewert, B. Pardo, M. Muller, and M. Plumbley, "Score-informed source separation for musical audio recordings: An overview," *IEEE Signal Process. Mag.*, vol. 31, no. 3, pp. 116–124, May 2014.
[21] J.-J. Aucouturier and F. Pachet, "Scaling up music playlist generation," in *Proc. IEEE Int. Conf. Multimedia Expo.*, Aug. 2002, vol. 1, pp. 105–108.
[22] T. Bertin-Mahieux, D. Eck, and M. Mandel, "Automatic tagging of audio: The state-of-the-art," in *Machine Audition: Principles, Algorithms and Systems*, W. Wang, Ed. Hershey, PA, USA: IGI Global, 2010.
[23] Y.-H. Yang and H. H. Chen, *Music Emotion Recognition*. Boca Raton, FL, USA: CRC, 2011.
[24] N. Collins, "Computational analysis of musical influence: A musicological case study using mir tools," in *Proc. ISMIR*, 2010, pp. 177–182.
[25] D. Schwarz, "Concatenative sound synthesis: The early years," *J. New Music Res.*, vol. 35, no. 1, pp. 3–22, Mar. 2006.
[26] L. Deng and D. Yu, *Deep Learning: Methods and Applications*. Delft, The Netherlands: Now, 2014.
[27] Y. Bengio, I. Goodfellow, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2015.
[28] F. Bastien *et al.*, "Theano: New features and speed improvements," in *Proc. Deep Learn. Unsupervised Feature Learn. NIPS 2012 Workshop*, 2012, pp. 1–10.
[29] J. Bergstra *et al.*, "Theano: A CPU and GPU math expression compiler," presented at the Python Sci. Comput. Conf., Austin, TX, USA, Jun. 2010.
[30] G. Montavon, G. B. Orr, and K.-R. Müller, *Neural Networks, Tricks of the Trade, Reloaded*, ser. Lecture Notes Comput. Sci. (LNCS 7700). New York, NY, USA: Springer, 2012.
[31] T. L. Li, A. B. Chan, and A. H. Chun, "Automatic musical pattern feature extraction using convolutional neural network," in *Proc. Int. Conf. Data Mining Appl.*, 2010, pp. 546–550.
[32] G. A. Wiggins, "Semantic gap?? Schemantic schmap!! Methodological considerations in the scientific study of music," in *Proc. IEEE Int. Symp. Mulitmedia*, Dec. 2009, pp. 477–482.
[33] J.-J. Aucouturier and F. Pachet, "Improving timbre similarity: How high is the sky?," *J. Negative Results Speech Audio Sci.*, vol. 1, no. 1, pp. 1–13, 2004.
[34] E. Pampalk, A. Flexer, and G. Widmer, "Improvements of audio-based music similarity and genre classification," in *Proc. Int. Soc. Music Info. Retrieval*, Sep. 2005, pp. 628–233.
[35] A. Flexer, "A closer look on artist filters for musical genre classification," in *Proc. ISMIR*, Sep. 2007, pp. 341–344.
[36] A. Flexer, D. Schnitzer, M. Gasser, and T. Pohle, "Combining features reduces hubness in audio similarity," in *Proc. Int. Symp. Music Info. Retrieval*, 2010, pp. 171–176.
[37] B. L. Sturm, "Classification accuracy is not enough: On the evaluation of music genre recognition systems," *J. Intell. Inf. Syst.*, vol. 41, no. 3, pp. 371–406, 2013.
[38] J. Urbano, M. Schedl, and X. Serra, "Evaluation in music information retrieval," *J. Intell. Inf. Syst.*, vol. 41, no. 3, pp. 345–369, Dec. 2013.
[39] B. L. Sturm, ""Horse" inside: Seeking causes of the behaviours of music content analysis systems," in *Proc. ACM Comput. Entertainment*, accepted for publication.
[40] B. L. Sturm, C. Kereliuk, and A. Pikrakis, "A closer look at deep learning neural networks with low-level spectral periodicity features," in *Proc. Int. Workshop Cognitive Inf. Process*, 2014, pp. 1–6.
[41] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proc. CVPR*, 2015, pp. 427–436.
[42] N. Dalvi, P. Domingos, S. Mausam, Sanghai, and D. Verma, "Adversarial classification," in *Proc. KDD*, 2004, pp. 99–108.
[43] N. Griffith and P. M. Todd, *Musical Networks: Parallel Distributed Perception and Performance*. Cambridge, MA, USA: MIT Press, 1999.
[44] C. J. C. Burges, J. C. Platt, and S. Jana, "Distortion discriminant analysis for audio fingerprinting," *IEEE Trans. Speech Audio Process.*, vol. 11, no. 3, pp. 165–174, May 2003.
[45] B. Matityaho and M. Furst, "Neural network based model for classification of music type," in *Proc. Conv. Elect. Electron. Eng. Israel*, Mar. 1995, pp. 1–5.
[46] N. Vempala and F. Russo, "Predicting emotion from music audio features using neural networks," in *Proc. CMMR*, 2012, pp. 336–343.
[47] B. Whitman, G. Flake, and S. Lawrence, "Artist detection in music with Minnowmatch," in *Proc. IEEE Workshop Neural Netw. Signal Process.*, Sep. 2001, pp. 559–568.
[48] G. Papadopoulos and G. Wiggins, "AI methods for algorithmic composition: A survey, a critical view and future prospects," in *Proc. AISB Symp. Musical Creativity*, 1999, pp. 110–117.
[49] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Trans. Speech Audio Process.*, vol. 10, no. 5, pp. 293–302, Jul. 2002.
[50] S. Dieleman, P. Brakel, and B. Schrauwen, "Audio-based music classification with a pretrained convolutional network," in *Proc. ISMIR*, 2011, pp. 669–674.
[51] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proc. ISMIR*, 2011, pp. 591–596.
[52] F. Weninger, F. Eyben, and B. Schuller, "On-line continuous-time music mood regression with deep recurrent neural networks," in *Proc. ICASSP*, May 2014, pp. 5412–5416.
[53] E. Battenberg and D. Wessel, "Analyzing drum patterns using conditional deep belief networks," in *Proc. ISMIR*, 2012, pp. 37–42.
[54] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Audio chord recognition with recurrent neural networks," in *Proc. ISMIR*, 2013, pp. 335–340.
[55] E. Humphrey and J. Bello, "From music audio to chord tablature: Teaching deep convolutional networks toplay guitar," in *Proc. ICASSP*, May 2014, pp. 6974–6978.
[56] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.
[57] C. N. Silla, A. L. Koerich, and C. A. A. Kaestner, "The Latin music database," in *Proc. ISMIR*, 2008, pp. 451–456.
[58] B. L. Sturm, "A survey of evaluation in music genre recognition," in *Adaptive Multimedia Retrieval: Semantics, Context, and Adaptation*, A. Nürnberger, S. Stober, B. Larsen, and M. Detyniecki, Eds. Zug, Switzerland: Springer, Oct. 2014, vol. LNCS 8382, pp. 29–66.
[59] J. B. Allen and L. Rabiner, "A unified approach to short-time Fourier analysis and synthesis," *Proc. IEEE*, vol. 65, no. 11, pp. 1558–1564, Nov. 1977.
[60] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. Berlin, Germany: Springer-Verlag, 2009.
[61] D. Griffin and J. S. Lim, "Signal estimation from modified short-time Fourier transform," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-32, no. 2, pp. 236–243, Apr. 1984.
[62] M. Henaff, K. Jarrett, K. Kavukcuoglu, and Y. LeCun, "Unsupervised learning of sparse features for scalable audio classification," in *Proc. Int. Soc. Music Inf. Retrieval*, Oct. 2011, pp. 681–686.
[63] B. L. Sturm, "An analysis of the GTZAN music genre dataset," in *Proc. ACM MIRUM Workshop*, Nov. 2012, pp. 7–12.
[64] M. Schedl, A. Flexer, and J. Urbano, "The neglected user in music information retrieval research," *J. Intell. Inf. Syst.*, vol. 41, no. 3, pp. 523–539, 2013.
[65] S. Dixon, F. Gouyon, and G. Widmer, "Towards characterisation of music via rhythmic patterns," in *Proc. ISMIR*, 2004, pp. 509–517.

Corey Kereliuk received the Ph.D. degree in music from McGill University, Montreal, QC, Canada, in 2013.

His research interests include signal processing and machine learning as they apply to the analysis, modification, and synthesis of music recordings.

Bob L. Sturm (S'06–M'09) received the B.A. degree in physics from the University of Colorado, Boulder, CO, USA, in 1998, the M.A. degree in music, science, and technology from Stanford University, Stanford, CA, USA, in 1999, and the M.S. degree in multimedia engineering and M.S. and Ph.D. degrees in electrical and computer engineering from the University of California at Santa Barbara (UCSB), Santa Barbara, CA, USA, in 2004, 2007, and 2009, respectively.

In 2009, he was a Chateaubriand Post-Doctoral Fellow with the Institut Jean Le Rond d'Alembert, Equipe Lutheries, Acoustique, Musique (LAM), Universit Pierre et Marie Curie, Paris, France. From 2010 to 2014, he was an Assistant and Associate Professor with the Department of Architecture, Design, and Media Technology, Aalborg University, Copenhagen, Denmark. In December 2014, he became a Lecturer in Digital Media with the School of Electronic Engineering and Computer Science, Queen Mary University, London, U.K. His research interests include audio and music signal processing, machine listening, and evaluation.

Dr. Sturm was the recipient of a two-year Independent Postdoctoral Grant from the Danish Agency for Science, Technology, and Innovation in 2011.

Jan Larsen (S'90–M'92–SM'03) received the M.Sc. and Ph.D. degrees in electrical engineering from the Technical University of Denmark (DTU), Kongens Lyngby, Denmark, in 1989 and 1994, respectively.

He is an Associate Professor of Digital Signal Processing with the Department of Applied Mathematics and Computer Science, DTU. He has authored or coauthored more than 150 papers and book chapters within the areas of machine learning, signal processing, and cognitive systems with application to audio, multimedia, sensor data, monitoring, biomedical, data/web-mining, and pattern recognition. According to Google Scholar, as of September 2015, his research impact is 3271 citations, h-index equal to 31, and i-10 index equal to 75.

Prof. Larsen is a Steering Committee Member of the Audio Signal Processing Network in Denmark (2006–present), an Editorial Board Member of *Signal Processing* (2006–2007), and Guest Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS, *Journal of VLSI Signal Processing Systems*, and *Neurocomputing*. He participated in conference organizations for the IEEE Workshop on Machine Learning for Signal Processing (formerly Neural Networks for Signal Processing) (1999–2015). He is Director of the Danish Sound Innovation Network (2009–present), Chair of the IEEE Denmark Sections Signal Processing Chapter (2002–present), and was previously Chair of the IEEE Machine Learning for Signal Processing Technical Committee of the IEEE Signal Processing Society (2005–2007). He has participated in more than ten national and international research and innovations programs and has served as reviewer for many international journals, conferences, publishing companies, and research funding organizations.