Training Deep Convolutional Networks with Unlimited Synthesis of Musical Examples for Multiple Instrument Recognition

Rameel Sethi, Noah Weninger, Abram Hindle, Vadim Bulitko, Michael Frishkopf*
Department of Computing Science and
Department of Music *
University of Alberta, Canada
{rameel,nweninge,hindle1,bulitko,michaelf}@ualberta.ca

ABSTRACT

Deep learning has yielded promising results in music information retrieval and other domains compared to machine learning algorithms trained on hand-crafted feature representations, but is often limited by the availability of data and vast hyper-parameter space. It is difficult to obtain large amounts of annotated recordings due to prohibitive labelling costs and copyright restrictions. This is especially true when the MIR task is low-level in nature such as instrument recognition and applied to wide ranges of world instruments, causing most MIR techniques to focus on recovering easily verifiable metadata such as genre. We tackle this data availability problem using two techniques: generation of synthetic recordings using MIDI files and synthesizers, and by adding noise and filters to the generated samples for data augmentation purposes. We investigate the application of deep synthetically trained models to two related low-level MIR tasks of frame-level polyphony detection and instrument classification in polyphonic recordings, and empirically show that deep models trained on synthetic recordings augmented with noise can outperform a majority class baseline on a dataset of polyphonic recordings labeled with predominant instruments.

1. INTRODUCTION

Music information retrieval (MIR) encompasses a wide range of tasks for classification or discovery of structure in music. These tasks may range from track-level descriptions such as genre and artist to audio analysis of frame-level information such as chord tagging and instrument activations. Polyphony estimation and instrument recognition are two related frame-level MIR tasks that may be improved by the use of deep neural networks due to their ability to learn representations without extensive domain knowledge of music, but are limited by a severe lack of recordings with frame-level annotations. We thus experiment with the use of synthetically generated recordings augmented with the addition of noise for training deep convolutional networks for these tasks.

The ability to search based on instrumentation is fundamental to several applications of music information retrieval such as ethnomusicology, efficient querying of music databases, and playlist recommendation systems. Services like Shazam [1] allow users to identify a currently playing track through input of a short segment of audio, as well as discover new tracks similar to the one currently playing. Interest in applying MIR techniques to field recordings has also been expressed, even though such research is still in an early stage [2].

Instrument recognition, among other MIR classification tasks, is largely dependent on the availability of large amounts of clean, annotated audio for training classifiers. Recently, deep neural networks have achieved state-of-the-art results in classification tasks in a number of domains [3]. Music information retrieval is no exception; deep learning has achieved state-of-the-art results in MIR tasks including predominant instrument recognition [4] and genre recognition [5]. The surge in popularity of deep learning approaches can be attributed to their flexibility and large number of parameters that may be learned; however, training deep models requires large amounts of labelled examples.

Although there are many datasets containing vast amounts of audio either on the web or in ethnomusicology archives in the form of field recordings, it is rare for these audio databases to be accompanied by adequate labels. There are a number of reasons for the lack of sufficient labelling, particularly in the case of musical attributes at the level of a single audio analysis frame–a small number of audio samples (e.g., 256). It is difficult for musicologists to label audio clips on a frame-level basis since each audio frame needs to be listened to before assigning labels. Additionally, many music databases are not open-access due to copyright restrictions.

Synthetic generation of training examples is one method of data augmentation that has found success in domains such as image classification [6]. This involves generation of realistic training samples using some prior knowledge of the underlying data distribution or other domain expertise. An important component of realism in data augmentation is the application of deformations, most prominently for images [7]. Augmentation techniques may similarly be applied to audio in the form of noise addition, as long as labels are either unchanged or transformed appropriately [8].

In this paper, we explore the use of synthetically gener-

ated audio recordings based on a predefined specification of instruments as training data for instrument recognition in polyphonic music. We generate multi-track compositions of instruments following General MIDI (GM) standard specification of instruments and render these to audio using a soundfont synthesizer. We apply deep convolutional neural networks trained on these synthetic recordings to the MIR tasks of polyphony detection and instrument recognition.

The main contribution of this work is that it demonstrates the effectiveness of deep classifiers, trained on synthetic generated examples, on MIR tasks where they share no training data yet still perform. This implies that existing MIR tools that employ deep learning can augment their training sets and improve their robustness with synthesized examples.

2. RELATED WORK

2.1 Polyphony Estimation and Instrument Recognition in Polyphonic Music

In this work we consider polyphony to be the number of instruments sounding, rather than multiple sounds from the same instruments, such as a 6-string guitar). Polyphony estimation in music is a challenging task due to overlap between sound sources, in this case instruments, in both time and frequency domains. Polyphony inference in music is often performed through the use of frame-based models for multiple fundamental frequency (f_0) estimation [9]. More recently, convolutional neural networks have been used for instrument recognition in polyphonic music [10]. Polyphonic instrument recognition is an instance of the more general problem of sound-source separation, where the task is to separate individual sources of audio from a given mixture. A classical approach to sound-source separation utilizes non-negative matrix factorization with sparse coding [11], while more recently deep networks have been used [12]. Our work seeks to model polyphony estimation and instrument recognition in tandem using a single classifier since they are intertwined in nature.

2.2 Deep Convolutional Neural Networks in MIR

Convolutional neural networks or convnets are artificial neural network architectures that use convolution in place of matrix multiplication in some layers. Deep neural networks, multilayered networks, have the ability to learn feature representations at multiple levels [13], which often results in improved performance over hand-crafted features and eliminates the need for area experts to spend large amounts of time and effort in finding effective feature representations. The success of deep learning has its origins in areas such as computer vision [14] [15] and speech recognition [16]. The use of deep learning has improved upon the state-of-the-art in several MIR tasks such as automatic music transcription [17] [18] music recommender systems [19]. Deep belief networks have also been used in conjunction with hidden Markov model frame-smoothing methods for tablature transcription [20]. In order to speed up training, and to yield benefits of transfer of knowledge from a source task to a target task, deep networks are often trained starting with pretrained weights [21]. Our aim in this paper is not to investigate novel approaches to improving performance of deep convolutional networks in MIR tasks, but rather to simply utilize existing networks while focusing on synthetic data augmentation aspects as described in later sections.

2.3 Data Augmentation and Synthetic Data Generation

Data augmentation is the transformation of existing data by adding noise or other means to make more training and test examples. Data augmentation can be used to help simulate noisy environments where a classifier is expected to run. Training on both clean and augmented or noisy data can improve the accuracy of a classifier to future unseen which we consider to be the quality of robustness. Thus, data augmentation is of utmost importance in producing a better model. Audio data augmentation utilizing label-preserving audio transformations including pitch shifting has been shown to improve classification accuracy on singing voice detection [22]. Recently, software frameworks for data augmentation [23] [8] approach data augmentation in music as application of a pipeline of transforms which may be label-invariant or label-variant in nature. Software frameworks for data augmentation in soundscape event classification have also been built [24]. However, there is often no access to large amounts of real annotated music recordings; augmentation using synthetic examples, such as generated music, may help in such cases. Synthetic data generation has been applied to pose recognition [25] and person re-identification [26]; in music it has been used to generate synthetic mixtures of monophonic instruments to improve performance in instrument transcription [27]. The goal of our approach is to complement and extend the use of data augmentation by training deep convolutional networks on synthetically generated clips of multiple instruments. This synthesis and augmentation enables us to better train and improve MIR task performance of supervised classifiers.

3. PROBLEM FORMULATION

In this section we formally define the twofold problem of polyphony estimation and instrument recognition in polyphonic music. There is currently no standard MIREX [28] task definition for instrument recognition largely due to the ambiguity between producing frame-level or track-level predictions. We use the definition of instrument recognition by of Li et al. [10], but modified to include polyphony estimation as well.

The input is an audio segment with a known maximum polyphony p (including the possibility of no instruments playing) and total number of possible instruments l according to an instrument taxonomy for the specification against which synthetic recordings are generated. The output is a vector $y \in \{0,1\}^{l+p+1}$, a concatenation of polyphony

estimation, and instrument estimation.

$$y_i = \begin{cases} \begin{cases} 1 & \text{if polyphony is } i \\ 0 & \text{otherwise} \end{cases} & 0 \leq i \leq p \\ \begin{cases} 1 & \text{if instrument } i-p-1 \\ & \text{is playing} \\ 0 & \text{otherwise} \end{cases} & p < i \leq l+p+1 \end{cases}$$

For example given $l = 5$ (5 instruments) and maximum polyphony of 3, if the last 3 of the 5 instruments were sounding the vector y would be the concatenation of $[0, 0, 0, 1]$ (3 instruments) and $[0, 0, 1, 1, 1]$ (the last 3 instruments are playing) to form the vector $[0, 0, 0, 1, 0, 0, 1, 1, 1]$. An output vector might look like $[0.1, 0.2, 0.3, 0.4, 0.0, 0.1, 0.9, 0.8, 0.7]$ which when processed with softmax on the polyphony would produce the expected output vector after normalizing. This is formulated as a multi-label classification problem since multiple instruments may be playing according to the maximum polyphony specified. If a prediction results in a mismatch between the predicted polyphony number and the number of instruments, only the prediction of polyphony is marked correct if the polyphony is correct and vice versa for instruments.

4. CONTINUOUS SYNTHESIS OF MIDI MUSIC TRAINING EXAMPLES

In this section we describe a simple algorithm for continuously training a polyphony estimation and instrument recognition classifier employing continuous synthesis of training examples as music represented as MIDI files.

4.1 Data Generation

To overcome the difficulty of finding large annotated datasets which cover a wide variety of musical styles, our system generates synthetic training data in the form of MIDI compositions of instrument note events sampled at random following a uniform distribution over the complete General MIDI specification (GM1) [29] of 128 program numbers. It is worth noting that not all program numbers correspond to instruments and thus some irrelevant classes exist in the resulting classifier as well as in the training data. The polyphony of each composition is chosen uniformly at random from the range [0, 3]. We choose this range since performance on polyphony tasks is found to decrease significantly as the maximum polyphony of the track under consideration increases beyond this range. We use the Python library *mingus* [1] to create MIDI files.

In the generation process, we first choose a subset of available instruments to include in the clip. The generated file contains one track for each instrument present. Notes are then placed in each track until a full 2 second bar is filled (120 BPM), with no overlapping of notes within the track. Notes range in pitch from C2 (MIDI 24, 65.41 Hz) to C7 (MIDI 84, 2093 Hz) and in duration from 1/4 to 1/16 of the clip. The note pitches were chosen to be close to the instruments with lowest and highest fundamental frequencies, the cello and piccolo respectively.

Algorithm 1: Pseudocode describing operation of synthesis loop

```
network: A polyphony/instrument classifier with
  learned weights
regen: Number of epochs between data generation
accuracy← 0
tolerance← 10^-4
learning-rate← 0.1
while Δaccuracy > tolerance do
    generate labelled music
    generate noised versions of music, using the
      randomized noise functions of described in
      Section 4.3
    combine both datasets
    learning-rate← 0.1e^(-0.01*epoch) + 0.0000001
    split into train and test sets
    train network for Regen epochs
    accuracy← evaluate performance
```

Once the MIDI files have been generated, they are rendered to WAV using *TiMidity++* [2] using the freely-available Fluid R3 GM soundfont [3]. Some of the files may be slightly longer than 2 seconds because the instruments have significant sustain; to remedy this, the WAV files are then clipped to exactly 2 seconds in length using ffmpeg [4]. If the audio is less than 2 seconds in playing time, then the clip is padded with silence.

4.2 Synthesis Loop

The synthesis loop focuses on generating new samples to train and test by generating music as described in the previous section, adding noise to the music, and then training and testing on the new music for some specified number of epochs, which we call $regen$ for the remainder of this paper. After $regen$ epochs, the neural network is evaluated and the music is thrown out, and the loop repeats. It should be noted that the the purpose of discarding old clips and regenerating new clips after $regen$ epochs is not to yield performance improvements on the test set, but rather to keep the process memory-efficient without having to save all clips on disk. The learning rate is exponentially scheduled to decay from a starting value of 0.1 as described below. Figure 1 shows a diagram of the operation of the synthesis loop.

4.3 Noise

Noise is added to the music with the use of csound [5], a program utilizing an audio domain-specific language (DSL).

[1] https://bspaans.github.io/python-mingus/

[2] http://timidity.sourceforge.net/

[3] https://packages.debian.org/stretch/fluid-soundfont-gm

[4] https://ffmpeg.org

[5] http://csound.github.io/

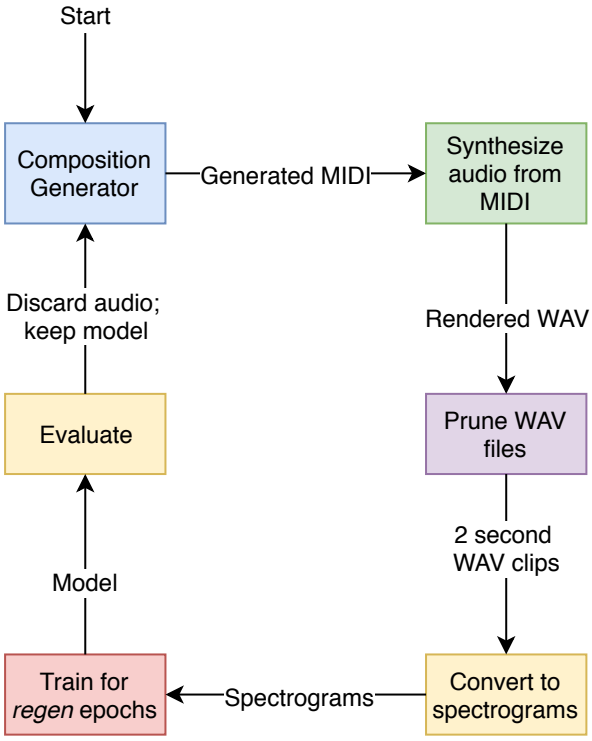


Figure 1: Diagram of synthesis loop

An effect from the following noise effects is chosen uniformly at random and applied to each sample. [6]

- reverb: random reverb added to the signal
- subtlereverb: a more subtle reverb added to the signal
- manytrees: a lowpass filter meant to emulate trees dampening sound
- tree: a small lowpass filter
- high pass: a high pass filter at a random frequency
- mid high pass: a high pass filter at a random frequency between 800 and 2000hz
- high high pass: a high pass filter at a random frequency above 2000hz
- hiss: lowpassed white noise added to the signal
- whitenoise: white noise added to the signal

4.4 Spectrogram Design and Input

The network input is a 227x227 (scaled down from 256x256 to fit AlexNet input dimensions without cropping) spectrogram image generated using the short-time Fourier transform (STFT). 256 overlapping FFTs of size 512 are taken over the 2 second synthesized audio clip using a Hamming window. We reduce the frequency range to between 100 and 10000 Hz to reduce noise; although the cello has a minimum frequency of 65 Hz, most notes played are above this frequency. Log-amplitudes with linear frequency buckets are used for spectrogram creation.

[6] The noise program we used: `https://github.com/abramhindle/audio-fuzzer`

There are 3 channels of input: amplitude, phase and phase gradient with respect to time. Supplying the network with the phase gradient is intended to enable more precise frequency estimation, in a manner inspired by the phase vocoder.

4.5 Network Architecture

We use the AlexNet [7] convolutional neural network architecture with weights pretrained on ImageNet [30]. The final softmax output layer is modified to have only $l+p+1$ output classes instead of the usual 1000 classes for ImageNet. A one-hot encoding is used for the polyphony class labels. The network output consists of two parts: first $n+1$ neurons, one-hot, indicating how many of the n instruments are sounding, then n neurons indicating the probability of each of the n instruments being present in the clip. The output is interpreted by choosing the k instruments with highest probability, given that k instruments are predicted to be sounding.

For all training during experiments a stochastic gradient descent (SGD) optimizer is used with learning rate determined according to the schedule used in Algorithm 1.

Hyperparameter and network selection is performed using a grid search over the following parameters (each repeated 5 times):

- Batch size: in range [5, 50, 500]
- Epochs: In range [10, 25, 50, 100, 200, 500, 1000, 2000]
- Loop: Whether we regenerate compositions continuously in a loop, or use a fixed set of compositions
- $regen$: In range [5, 10, 20, 50]
- Networks: One of [AlexNet, VGG-19 [31], GoogleNet [32]]

Figure 3 shows a partial plot of the grid search described above.

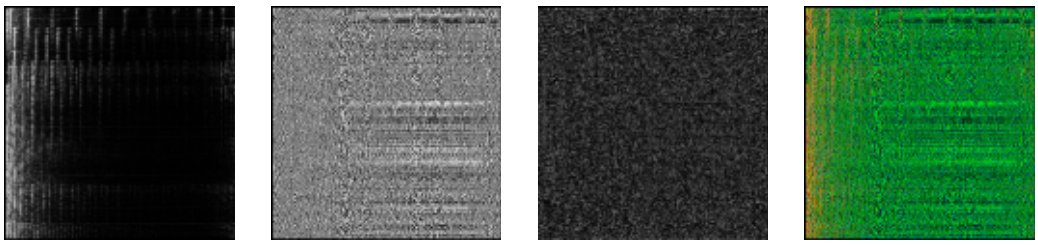
Based on the best grid search performance, for all networks discussed in this paper we use parameters of batch size of 5, with a maximum of 100 epochs, using the AlexNet architecture with $regen = 10$. We also utilize early stopping with a loss tolerance of 10^{-4} and a patience of 3 epochs.

5. EXPERIMENTS

5.1 Evaluation Criteria

The evaluation metrics used on our polyphony/instrument classification experiments are:

- Polyphony Accuracy: The polyphony accuracy is the percent of examples for which the correct number of sounding instruments was predicted. Polyphony follows a one-hot encoding scheme ranging from zero to the maximum number of instruments sounding simultaneously, yielding a binary crossentropy loss function.



(a) Amplitude (b) Phase (c) Phase gradient (d) Resulting spectrogram

Figure 2: RGB spectrogram composed of amplitude, phase and phase gradient channels

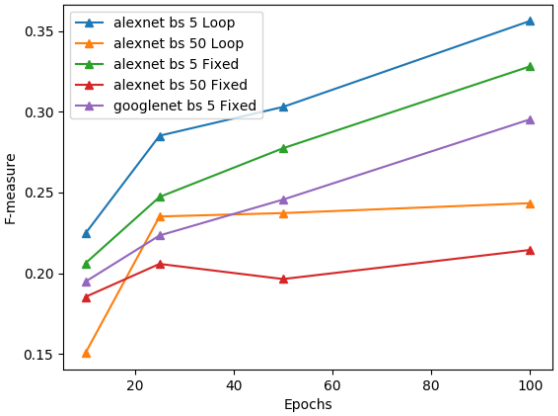


Figure 3: Plot of f-measure against number of epochs for different architectures, batch sizes and continuous versus fixed composition generation

- Instrument Accuracy: The instrument accuracy is the percent of examples for which the correct instruments were guessed. Instrument prediction is treated as a multilabel classification problem, necessitating a binary crossentropy loss function.
- Total Accuracy: Determined through a binary crossentropy loss function over both polyphony and instruments.
- Confusion Matrices: These are computed for classification of both polyphony and instruments.
- F-measure: The F-measure is required for polyphony and instrument recognition in music evaluation due to class imbalances in frame-level polyphony and number of instruments.

5.2 Synthetic Test Performance

Our first task is to classify the polyphony and the instruments sounding of our generated music. Per each model, we perform 8 runs with 10,000 training samples (generated at random for each run), 2,500 validation samples, and 2,500 test samples generated once and held out. We set the polyphony $p = 3$ and total instruments $l = 127$. The samples are generated in such a way that the numbers of classes of each polyphony are the same; there may be slight differences in the numbers of samples containing each instrument, but not enough to cause a statistically significant class imbalance problem since each instrument is equally likely to be chosen. The validation and test sets were kept constant across trials. For all experiments, we evaluate on the validation set at the end of each epoch to check for eventual overfitting within a tolerance level of 10^{-4}.

To investigate the robustness of synthetically trained networks to noise, we evaluate models obtained from spectrograms generated from clean WAV audio on test datasets with noisy audio and vice versa, where the noise added can be at random from any of the types described previously. The number of samples trained on is kept constant regardless of whether the model is trained purely on clean data or a mix of clean and noisy data (where the ratio of clean to noisy data is 1-1). The results are summarized in Table 1.

Table 2 summarizes polyphony F-1 scores obtained on a separately generated test set of 2500 MIDI samples for both clean and noisy models. For polyphony values of 1 and 2, clean models show poor performance when applied to noisy data and vice versa. This may be due to the additional noise effect being modeled as a sound source.

Despite the large number of output classes, both polyphony estimation and instrument recognition outperform a majority class baseline on a synthetically generated test set. As expected, models trained on clean audio perform best on clean test audio. Surprisingly, however, clean audio models applied to noisy data perform significantly better than noisy audio models applied to clean data. The clean model did suffer much in terms of performance by being applied to noisy samples, while noisy data demonstrated more stability or robustness by maintaining performance on clean and noise evaluations although at generally lower performance.

5.3 Evaluation on real world musical recordings

In order to demonstrate the feasibility of this method on actual recordings we engage in polyphony detection (the number of instruments sounding) and instrument recognition on existing benchmark datasets. We evaluate our synthesis loop algorithm on the IRMAS [33] dataset 2874 test samples in the form of excerpts from songs ranging from 5 to 20 seconds in length containing a total of 11 instruments. We do not train on any of the IRMAS samples. In order to make the classification on spectrograms resulting from clips longer than 2 seconds a well-defined task, we divide the clip into 2-second overlapping windows

Train-Test	Poly. Prec.	Poly. Recall	Poly. F-1	Inst. Prec.	Inst. Recall	Inst. F-1
Clean-Clean	$\mathbf{0.73 \pm 0.02}$	$\mathbf{0.73 \pm 0.02}$	$\mathbf{0.73 \pm 0.02}$	$\mathbf{0.35 \pm 0.02}$	$\mathbf{0.33 \pm 0.02}$	$\mathbf{0.34 \pm 0.02}$
Noisy-Noisy	0.53 ± 0.02	0.53 ± 0.02	0.53 ± 0.02	0.25 ± 0.02	0.21 ± 0.02	0.23 ± 0.02
Clean-Noisy	$\mathbf{0.65 \pm 0.03}$	$\mathbf{0.65 \pm 0.03}$	$\mathbf{0.65 \pm 0.03}$	$\mathbf{0.28 \pm 0.03}$	$\mathbf{0.24 \pm 0.03}$	$\mathbf{0.26 \pm 0.03}$
Noisy-Clean	0.52 ± 0.03	0.52 ± 0.03	0.52 ± 0.03	0.24 ± 0.03	0.21 ± 0.03	0.23 ± 0.03

Table 1: Test precision, recall and F-1 scores of clean and noisy models

Train-Test	Polyphony	F-1
Clean-Clean	0	$\mathbf{0.99 \pm 0.06}$
	1	$\mathbf{0.80 \pm 0.06}$
	2	$\mathbf{0.60 \pm 0.05}$
	3	0.36 ± 0.08
Noisy-Noisy	0	$\mathbf{0.99 \pm 0.06}$
	1	$\mathbf{0.99 \pm 0.06}$
	2	$\mathbf{0.99 \pm 0.06}$
	3	$\mathbf{0.99 \pm 0.06}$
Clean-Noisy	0	0.96 ± 0.04
	1	0.15 ± 0.07
	2	0.01 ± 0.08
	3	0.85 ± 0.06
Noisy-Clean	0	0.99 ± 0.07
	1	0.15 ± 0.06
	2	0.01 ± 0.01
	3	$\mathbf{0.95 \pm 0.04}$

Table 2: Polyphony F-1 scores on test MIDI

with 50 per cent hop size and evaluate our synthesis loop model on each window. We calculate the mean frame-level F-score per instrument over all recordings. For IRMAS, the labels are predominant instruments, but since multiple instruments are labeled per recording we may use the labels for multiple instrument recognition purposes. We performed a manual mapping from the instrument taxonomy of the dataset to the MIDI specification. We count a prediction as correct if any of the MIDI instruments in the original instrument's mapping are predicted. The mean polyphony and instrument recognition scores along with standard deviation over 8 trials are found using the same 8 models trained on clean synthetic compositions as described in the previous section. The noisy models did not perform as well on the IRMAS dataset. The results are summarized in Tables 3 and 4.

Poly	M	Precision	Recall	F-1
1	C	$\mathbf{0.50 \pm 0.05}$	$\mathbf{0.56 \pm 0.04}$	$\mathbf{0.53 \pm 0.05}$
	N	0.24 ± 0.06	0.10 ± 0.06	0.14 ± 0.06
2	C	$\mathbf{0.37} \pm 0.06$	$\mathbf{0.24 \pm 0.04}$	$\mathbf{0.29 \pm 0.06}$
	N	0.10 ± 0.05	0.05 ± 0.06	0.06 ± 0.05
3	C	$\mathbf{0.75 \pm 0.03}$	0.14 ± 0.05	0.24 ± 0.06
	N	$\mathbf{0.75 \pm 0.07}$	$\mathbf{0.28 \pm 0.05}$	$\mathbf{0.41 \pm 0.07}$
Mean	C	$\mathbf{0.48 \pm 0.05}$	0.28 ± 0.06	$\mathbf{0.35 \pm 0.06}$
	N	0.41 ± 0.08	$\mathbf{0.29 \pm 0.09}$	0.34 ± 0.05

Table 3: Test polyphony precision, recall and F-1 scores on IRMAS. N for noisy models, C for clean models

Polyphony estimation results from our synthetic classifier

Instrument	M	Precision	Recall	F-1
Cello	C	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	N	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Clarinet	C	$\mathbf{0.20 \pm 0.03}$	$\mathbf{0.01 \pm 0.01}$	$\mathbf{0.02 \pm 0.02}$
	N	0.02 ± 0.01	$\mathbf{0.01 \pm 0.01}$	$\mathbf{0.02 \pm 0.01}$
Flute	C	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	N	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Guitar (ac)	C	$\mathbf{0.04 \pm 0.01}$	$\mathbf{0.01 \pm 0.01}$	$\mathbf{0.03 \pm 0.01}$
	N	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Guitar (el)	C	$\mathbf{0.02 \pm 0.01}$	$\mathbf{0.01 \pm 0.01}$	$\mathbf{0.02 \pm 0.01}$
	N	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Organ	C	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	N	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Piano	C	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	N	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Saxophone	C	$\mathbf{0.02 \pm 0.01}$	$\mathbf{0.01 \pm 0.01}$	$\mathbf{0.02 \pm 0.01}$
	N	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Trumpet	C	$\mathbf{0.82 \pm 0.20}$	$\mathbf{0.07 \pm 0.04}$	$\mathbf{0.13 \pm 0.06}$
	N	0.05 ± 0.02	0.03 ± 0.01	0.04 ± 0.02
Violin	C	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	N	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Voice	C	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	N	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00

Table 4: Test instrument precision, recall and F-1 scores on IRMAS. N for Noisy Models, C for Clean Models

outperform a majority classifier, like ZeroR, at the frame level. The clarinet and trumpet classes yield the best average frame-level precision, while others such as flute fail to be positively classified. F-measure are not competitive with state-of-the-art approaches, Slizovskaia et al. [34] achieved 0.67 F-measure on IRMAS over all instruments. But the majority classifier is still outperformed for several instruments without any training on the IRMAS training set. Poor performance on some classes like voice may be explained by the lack of a standard MIDI program number for voice events. While instrument recognition is not much better than a majority class rule, this may again be attributed to the large number of output classes the model was trained on.

6. CURRENT LIMITATIONS AND FUTURE WORK

Our synthesis loop algorithm currently generates training examples at random. We could focus more on achieving high performance on a specific dataset (e.g. different genre) and tune the loop to generate data more similar to the test dataset under consideration.

Although we evaluated on a dataset of real music recordings, the effects of noise were not investigated. Augmenting these datasets with noise would enable us to further

test robustness of our synthesis loop algorithm to noise. An additional path to explore is comparison of our noising framework with other open-source frameworks utilizing both label-invariant and label-variant transformations [8] to investigate which types of noise effects and other augmentations yield the best improvements in performance.

It is worth noting that soundfonts such as the one used to render are mostly restricted to Western instruments and are typically rendered in 12-tone equal temperament. A different approach may need to be taken for non-Western music collections.

We restricted the use of transfer learning in this work to the use of pretrained model weights on ImageNet for facilitating speedup in network training. Transfer learning, the application of a model trained on one domain applied to a different domain, has been shown to improve performance between related MIR tasks such as genre classification and music similarity when faced with a limited number of training examples [35]. Transfer learning has also been applied to cross-domain prediction between music and speech [36], where it is shown that models trained on speech emotion generalize well to detection of music emotion and vice versa. One possible investigation is to fine-tune a synthetically trained model on real music to determine whether similar performance may be achieved in a sample-efficient manner.

7. CONCLUSIONS

In this paper we presented a method for overcoming limitations in the amount of training data available for MIR in the form of a simple synthesis loop algorithm for generating batches of synthetic music clips to be used in training a convolutional neural network. This network was to estimate the number of instruments sounding, as well as which instruments are sounding.

We show that our method results in reasonable performance on the tasks of polyphony estimation and instrument recognition in music without usage of any training data supplied by 3rd parties or the target task. The benefit of this work is that synthesis and augmentation lets classifiers perform on unseen data well, as demonstrated by clean versus noisy tasks, and as demonstrated by the real world examples. This implies that synthesis and augmentation has a place in MIR, as it opens up the possibility for training synthetically trained classifiers to achieve state-of-the-art performance in a variety of MIR tasks with limited training data available. We show that purely synthetic examples can achieve reasonable performance and robustness. Perhaps a combination of augmentation and synthesis can aid future MIR tasks?

We suggest some improvements to our algorithm to improve learning efficiency and bring performance closer to existing state-of-the-art methods. Future directions include studying generation of more realistic synthetic recordings, investigation of the effects of audio deformations other than noise by comparing different open-source audio deformation frameworks, and the possibility of using transfer learning to enable sample-efficient fine-tuning of trained deep audio classifiers.

Acknowledgments

This work was supported by a KIAS cluster grant. Some equipment (GPUs) used in this research were donated by the NVIDIA Corporation. Vadim Bulitko and Abram Hindle are supported by NSERC Discovery Grants.

8. REFERENCES

[1] A. Wang, "The shazam music recognition service," *Communications of the ACM*, vol. 49, no. 8, pp. 44–48, 2006.

[2] G. Tzanetakis, "Computational ethnomusicology: a music information retrieval perspective." in *ICMC*, 2014.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[4] Y. Han, J. Kim, K. Lee, Y. Han, J. Kim, and K. Lee, "Deep convolutional neural networks for predominant instrument recognition in polyphonic music," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 25, no. 1, pp. 208–221, 2017.

[5] K. Choi, G. Fazekas, and M. Sandler, "Automatic tagging using deep convolutional neural networks," *arXiv preprint arXiv:1606.00298*, 2016.

[6] P. Rajpura, M. Goyal, R. Hegde, and H. Bojinov, "Dataset augmentation with synthetic images improves semantic segmentation," *arXiv preprint arXiv:1709.00849*, 2017.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[8] B. McFee, E. J. Humphrey, and J. P. Bello, "A software framework for musical data augmentation." in *ISMIR*. Citeseer, 2015, pp. 248–254.

[9] C. Yeh, A. Roebel, and X. Rodet, "Multiple fundamental frequency estimation and polyphony inference of polyphonic music signals," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1116–1126, 2010.

[10] P. Li, J. Qian, and T. Wang, "Automatic instrument recognition in polyphonic music using convolutional neural networks," arXiv, Tech. Rep., November 2015, arXiv:1511.05520 [Cs]. [Online]. Available: http://arxiv.org/abs/1511.05520

[11] T. Virtanen, "Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria," *IEEE transactions on audio, speech, and language processing*, vol. 15, no. 3, pp. 1066–1074, 2007.

[12] G. Naithani, G. Parascandolo, T. Barker, N. H. Pontoppidan, and T. Virtanen, "Low-latency sound source separation using deep neural networks," in *Signal*

and Information Processing (GlobalSIP), 2016 IEEE Global Conference on. IEEE, 2016, pp. 272–276.

[13] G. E. Hinton, "Learning multiple layers of representation," *Trends in Cognitive Sciences*, vol. 11, no. 10, pp. 428–434, 2007.

[14] G. E. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[15] H. Lee, R. Grosse, R. Ranganath, and A. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the International Conference on Machine Learning*, Montréal, QC, 2009, pp. 609–616.

[16] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[17] E. J. Humphrey and J. P. Bello, "From music audio to chord tablature: Teaching deep convolutional networks toplay guitar," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 6974–6978.

[18] S. Sigtia, E. Benetos, N. Boulanger-Lewandowski, T. Weyde, A. d'Avila Garcez, and S. Dixon, "A hybrid recurrent neural network for music transcription," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2015, pp. 2061–2065.

[19] A. Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," *Advances in Neural Information Processing Systems*, no. 26, pp. 2643–2651, 2013.

[20] G. D. Burlet, "Guitar tablature transcription using a deep belief network," Ph.D. dissertation, University of Alberta, 2015.

[21] A. Van Den Oord, S. Dieleman, and B. Schrauwen, "Transfer learning by supervised pre-training for audio-based music classification," in *Conference of the International Society for Music Information Retrieval (ISMIR 2014)*, 2014.

[22] J. Schlüter and T. Grill, "Exploring data augmentation for improved singing voice detection with neural networks." in *ISMIR*, 2015, pp. 121–126.

[23] M. Mauch, S. Ewert *et al.*, "The audio degradation toolbox and its application to robustness evaluation," in *ISMIR*, 2013.

[24] J. Salamon, D. MacConnell, M. Cartwright, P. Li, and J. P. Bello, "Scaper: A library for soundscape synthesis and augmentation," in *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2017 IEEE Workshop on*. IEEE, 2017, pp. 344–348.

[25] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time human pose recognition in parts from single depth images," *Communications of the ACM*, vol. 56, no. 1, pp. 116–124, 2013.

[26] I. B. Barbosa, M. Cristani, B. Caputo, A. Rognhaugen, and T. Theoharis, "Looking beyond appearances: Synthetic training data for deep cnns in re-identification," *arXiv preprint arXiv:1701.03153*, 2017.

[27] H. Kirchhoff, S. Dixon, and A. Klapuri, "Multi-template shift-variant non-negative matrix deconvolution for semi-automatic music transcription," in *ISMIR*, 2012.

[28] X. Hu, J. H. Lee, D. Bainbridge, K. Choi, P. Organisciak, and J. S. Downie, "The mirex grand challenge: A framework of holistic user-experience evaluation in music information retrieval," *Journal of the Association for Information Science and Technology*, vol. 68, no. 1, pp. 97–112, 2017.

[29] midi.org. (2016, mar) General midi (gm 1). [Online]. Available: https://www.midi.org/specifications/item/general-midi

[30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

[31] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[32] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 1–9.

[33] J. J. Bosch, J. Janer, F. Fuhrmann, and P. Herrera, "A comparison of sound segregation techniques for predominant instrument recognition in musical audio signals." in *ISMIR*, 2012, pp. 559–564.

[34] O. Slizovskaia, E. Gómez Gutiérrez, and G. Haro Ortega, "Automatic musical instrument recognition in audiovisual recordings by combining image and audio classification strategies," in *Proceedings SMC 2016. 13th Sound and Music Computing Conference*. Zentrum für Mikrotonale Musik und Multimediale Komposition (ZM4), Hochschule für Musik und Theater Hamburg, 2016.

[35] P. Hamel, M. E. P. Davies, K. Yoshii, and M. Goto, "Transfer learning in mir: Sharing learned latent representations for music audio classification and similarity," in *14th International Conference on Music Information Retrieval (ISMIR '13)*, 2013.

[36] E. Coutinho, J. Deng, and B. Schuller, "Transfer learning emotion manifestation across music and speech," in *Neural Networks (IJCNN), 2014 International Joint Conference on*. IEEE, 2014, pp. 3592–3598.