SCAPER: A LIBRARY FOR SOUNDSCAPE SYNTHESIS AND AUGMENTATION

Justin Salamon[1,2*], *Duncan MacConnell*[1], *Mark Cartwright*[1], *Peter Li*[1], *Juan Pablo Bello*[1]

[1] Music and Audio Research Laboratory, New York University, NYC, NY, USA
[2] Center for Urban Science and Progress, New York University, NYC, NY, USA
[*]Please address correspondence to Justin Salamon (justin.salamon@nyu.edu)

ABSTRACT

Sound event detection (SED) in environmental recordings is a key topic of research in machine listening, with applications in noise monitoring for smart cities, self-driving cars, surveillance, bioacoustic monitoring, and indexing of large multimedia collections. Developing new solutions for SED often relies on the availability of strongly labeled audio recordings, where the annotation includes the onset, offset and source of every event. Generating such precise annotations manually is very time consuming, and as a result existing datasets for SED with strong labels are scarce and limited in size. To address this issue, we present Scaper, an open-source library for soundscape synthesis and augmentation. Given a collection of isolated sound events, Scaper acts as a high-level sequencer that can generate multiple soundscapes from a single, probabilistically defined, "specification". To increase the variability of the output, Scaper supports the application of audio transformations such as pitch shifting and time stretching individually to every event. To illustrate the potential of the library, we generate a dataset of 10,000 soundscapes and use it to compare the performance of two state-of-the-art algorithms, including a breakdown by soundscape characteristics. We also describe how Scaper was used to generate audio stimuli for an audio labeling crowdsourcing experiment, and conclude with a discussion of Scaper's limitations and potential applications.

Index Terms— Soundscape, synthesis, sound event detection.

1. INTRODUCTION

Sound event detection (SED) is the task of automatically identifying the source and location in time of different sounds as they occur in a continuous audio stream. The task has received growing interest from the research community over the last few years, with applications in noise monitoring for smart cities [1], bioacoustic species and migration monitoring [2], self-driving cars [3], surveillance [4] and large-scale multimedia indexing [5]. Training supervised algorithms to perform this task requires large, annotated datasets. The required precision of the annotations depends on the model: while there has been some recent work on training models from weakly labeled data (i.e. annotations that specify the presence/absence of a source in an audio recording without specifying its temporal location) [6], the majority of models proposed to date for SED require strongly labeled data, i.e. annotations that specify not only the source but also the onset and offset of every sound event. Even models that can be trained on weakly labeled data require strongly labeled data for evaluating their performance at finer temporal resolutions.

Generating such annotations is a laborious and time-consuming task, and consequently manually annotated datasets for SED with strong labels are very limited in size (e.g. the TUT Sound Events 2016 development set is 78 minutes long). For model training, one solution is data augmentation, i.e. the transformation/manipulation of existing training data in order to generate new labeled samples [7, 8, 9]. For SED, since the training data is comprised of soundscapes that contain multiple sound events, augmentations applied to the soundscape as a whole can certainly be helpful, but are limited in that the characteristics of the soundscape such as event timing, degree of event overlap, and signal-to-noise ratio (SNR) will remain unchanged even after transformation. For evaluation, a limitation of datasets based on real recordings is that it is not possible to control for different acoustic characteristics, which could be helpful in providing insight into the differences between different SED models.

To address these limitations, we present Scaper[1], an open-source Python library for soundscape synthesis and augmentation. Given a soundbank (collection) of isolated sound events, Scaper acts as a high-level, probabilistically controlled, audio sequencer that can generate new soundscapes or add sound events to existing ones while controlling characteristics such as the number and types of events, their timing, duration and SNR with respect to a "background" track. While there is a large body of research on problems related to soundscape synthesis such as binaural or spatial scene synthesis [10], acoustic event synthesis [11] and texture synthesis [12], such systems are typically not designed with the goal of training/evaluating machine (and human) SED performance. This means they are not necessarily designed to reproduce the types of acoustic scenes such models are likely to be evaluated on, they do not generate annotations that match the synthesized audio, and often they are not designed for batch processing and integration with machine learning pipelines. To the best of our knowledge, the only system designed specifically with SED in mind was the one proposed by Lafay et al. [13]. Scaper varies from this system in several ways: most importantly, while the aforementioned system only provides high-level controls for generating soundscapes based on fixed distributions, Scaper uses the concept of an "event specification" coupled with multiple distributions to provide a flexible level of control ranging from a high-level probabilistic soundscape definition down to specifying every detail of every sound event. Furthermore, Scaper supports applying audio transformations such as pitch shifting and time stretching individually to each sound event, significantly increasing the possible range and variability of the generated soundscapes. The library generates annotations in JAMS format [14] which stores Scaper-related metadata, facilitating a complete reconstruction of the soundscape from its JAMS annotation as well as supporting manipulation of the JAMS annotation to generate

[*]This work was partially supported by NSF awards 1544753 and 1633259, and a Google Faculty Award.

[1] https://github.com/justinsalamon/scaper

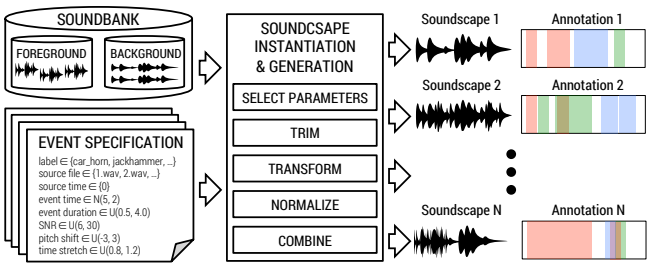

Figure 1: Block diagram of the Scaper synthesis pipeline.

variants of existing soundscapes. Finally, Scaper is implemented in Python, which means it does not require any proprietary software (such as, e.g., Matlab) and is easy to integrate with popular machine (and deep) learning Python libraries such as scikit-learn [15], TensorFlow [16] and Keras [17], as well as popular audio analysis Python libraries such as Essentia [18] and Librosa [19].

In the next section we provide an overview of Scaper, including design choices and functionality. Next, we demonstrate the utility of the library through a series of experiments: first, we use Scaper to generate a large dataset of urban soundscapes and evaluate state-of-the-art SED algorithms, including a breakdown by soundscape characteristics. Next, we describe how Scaper was used to generate audio stimuli for a crowdsourcing experiment on the accuracy of human sound event annotations as a function of sound visualization and soundscape characteristics. The paper concludes with a discussion of limitations and suggestions for new use cases.

2. SCAPER

SED is based on the notion that sounds in a soundscape can be broadly grouped into two categories: foreground sound events which are salient and recognizable, and background sounds, often regarded as a single holistic sound which is more distant, ambiguous, and texture-like [20, 21, 22]. Scaper was designed with the same paradigm in mind: a soundscape is generated as the summation of foreground events and a background recording. It is up to the user to curate a soundbank (collection) of their choice and organize the sounds into foreground and background folders, with a sub-folder for each sound class (label). As such, Scaper is content-agnostic and can be readily applied to a variety of audio domains including urban and rural soundscapes, bioacoustic recordings, indoor environments (e.g. smart homes) and surveillance recordings. A block diagram of Scaper's synthesis pipeline is given in Figure 1.

A key building block of Scaper is the *event specification*. An event specification stores all properties of a sound event that Scaper can control, namely: the event *label* (class), *source file* (i.e. the specific sound clip to be used), the *event duration*, the *source time* (i.e. when the event starts in the source clip), the *event time* (when the event should start in the generated soundscape), the *SNR* with respect to the background recording, the event *role* (foreground or background), *pitch shift* (in semitones, does not affect duration) and *time stretch* (as a factor of the event duration, does not affect pitch). Thus, a soundscape is defined by a set of event specifications, which are grouped into a foreground specification (for all foreground events) and a background specification. To define a soundscape, the user specifies a desired soundscape duration, a reference loudness level for the background, and then adds event specifications. For every property in an event specification the user provides a *distribution tuple*, which defines a distribution to sample the property value from. The distributions currently supported include *const* (specifying a constant value), *choose* (randomly selecting from a discrete list of values), *uniform*, *normal* and *truncnormal* (sampling from a continuous distribution), and additional distributions can be easily added. As such, the user has control over how detailed the specification is: from precisely defining every property of every event using constants to a high-level probabilistic specification that only specifies a distribution to sample from for every event property. Given the foreground and background specifications, the user can generate infinitely many soundscape instantiations[2].

An instantiated specification (i.e. with concrete values that have been sampled for all properties) is then used as a recipe for generating the soundscape audio, where all audio processing is performed using pysox [23]. One aspect of the generation that requires special care is the handling of SNR values. In particular, simple peak normalization does not guarantee that two sounds normalized to the same level will be *perceived* as equally loud. To circumvent this, Scaper uses Loudness Units relative to Full Scale (LUFS) [24], a standard measure of perceived loudness used in radio, television and Internet broadcasting. Thus, if an event is specified to have an SNR of 6, it means it will be 6 LUFS above the background level. Finally, Scaper saves the soundscape annotation in two formats: the first is a simple space-separated text file with three columns for the onset, offset and label of every sound event. This format is useful for quickly inspecting the events in a soundscape and can be directly loaded into software such as Audacity to view the labels along with the audio file. The second format is JAMS [14], originally designed as a structured format for music annotations, which supports storing unlimited, structured, file metadata. Scaper exploits this to store both the probabilistic and instantiated specifications of every sound event. This means that (assuming one has access to the original soundbank) Scaper can fully reconstruct the audio of a soundscape from its JAMS annotation. Scaper is open-source (see footnote 1) and we encourage contributions from the community to improve the library and implement new features.

3. THE URBAN-SED DATASET

To illustrate the utility of Scaper, we used it to generate a large dataset of 10,000 ten-second soundscapes for training and evaluating SED algorithms. We used the clips from the UrbanSound8K dataset [25], approximately 1000 per each of ten urban sound sources (each clip contains one of the ten sources), as the soundbank. UrbanSound8K is pre-sorted into 10 stratified folds, and so we use folds 1–6 for generating 6000 training soundscapes, 7–8 for generating 2000 validation soundscapes and 9–10 for generating 2000 test soundscapes. Soundscapes were generated using the following protocol: first, we add a background sound normalized to -50 LUFS. We use the same background audio file for all soundscapes, a 10 second clip of Brownian noise, which resembles the typical "hum" often heard in urban environments. By using a purely synthesized background we are guaranteed that it does not contain any spurious sound events that would not be included in the annotation. Next, we choose how many events to include from a discrete uniform distribution between 1–9. Every event is added with the same high-level specification: the label is chosen randomly from all 10 available sound classes, and the source file is chosen randomly from all clips matching the selected label. The source time is always 0, to ensure we do not miss the onset of an event. The start

[2]For an example see: `https://git.io/v9GGn`

time of the event in the generated soundscape is sampled from one of three distributions: uniform between 0–10, unimodal (normal with mean 5 and standard deviation of 2) and bimodal (two normals with means of 3 and 7 and a standard deviation of 2). By using these distributions for the event start times we obtain a variety of soundscapes, some in which the events are spread out and others in which they tend to be more clustered in time, consequently leading to a greater degree of overlap, henceforth referred to as polyphony. We define the maximum polyphony of a soundscape to be the greatest sound event polyphony observed at any point in time in the soundscape. The maximum polyphony is automatically computed by Scaper during generation and stored in the JAMS annotation. This will allow us to easily evaluate model performance as a function of maximum polyphony. Duration is chosen from a uniform distribution between 0.5–4 seconds (all clips in UrbanSound8K are at most 4 seconds), unless the source clip is shorter in which case the duration of the source clip is used. The SNR is sampled uniformly between 6–30 dB. Finally, every event is pitch shifted by a semitone amount sampled from a (continuous) uniform distribution between -3 and 3, and time stretched by a factor sampled from a uniform distribution between 0.8 and 1.2. For an example of implementing this procedure using Scaper see footnote 2. The resulting dataset, URBAN-SED, consists of 10,000 soundscapes totaling almost 30 hours with close to 50,000 annotated sound events with maximum polyphonies between one and seven. This makes it the largest strongly labeled dataset available for SED, though we could of course make it arbitrarily larger or smaller. The second largest, TUT-SED synthetic 2016 (also synthesized) [26], is roughly 9.5 hours long and to the best of our knowledge was generated using ad-hoc scripts which are not publicly available. URBAN-SED is made freely available online[3], and for reproducibility all scripts used to generate it, as well as run the machine learning experiments reported in the following section, are also made available online[4].

4. SCAPER FOR MACHINE LEARNING

For the machine learning experiments we evaluated two state-of-the-art models: the first is the Convolutional Recurrent Neural Network (CRNN) proposed by Cakir et al. [26]. We use the architecture identified by the authors to perform best on the TUT-SED-2016 development dataset [26], with 743k parameters (see [26] for details), and ensured our implementation was correct by training/testing it on that dataset, for which we obtained near-identical results to those reported in the paper. The second model is an adaptation of the Convolutional Neural Network (CNN) proposed by Salamon and Bello [27]. The original model was proposed for multi-class classification, and so to adapt it for multi-label classification we replace the final softmax activation with sigmoid activations. The original model had 241k parameters, and so to match the capacity of the CRNN we increase the number of convolutional filters to 64 in each layer. We also add batch normalization [28] to the output of the convolutional layers. Since we want to use the model for SED, we reduce the duration of the input representation to 1 s, and reduce the max pooling after the convolutional layers to (2,2). The resulting model has 720k parameters. An important difference between the models is that the CRNN outputs predictions at the frame level (e.g. 20 ms resolution) whereas the CNN outputs predictions at a 1 s temporal resolution. This is motivated by the fact that a 1 s temporal resolution would be sufficient for urban sound monitoring applications, has been used in recent work with promising results [5], and results in a model that is significantly faster to train. Both models are implemented in Keras [17] and trained using the Adam optimizer [29] with binary cross-entropy loss over 300 epochs with an early-stopping criterion of 100 epochs with no improvement to the segment-based F-measure [30], computed at a 1 s temporal resolution[5] using sed_eval [30] where an epoch represents one full pass over the training set. The training set contains 6000 soundscapes, and the validation and test sets contain 2000 soundscapes each.

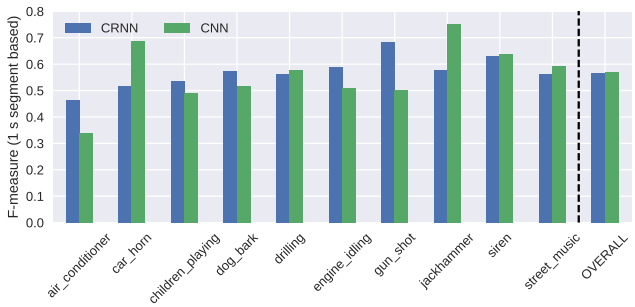

Figure 2: Performance (1 s segment-based F-measure) for the compared models (CRNN and CNN), by sound event class and overall.

The results are presented in Figure 2, in which we also include a breakdown of the performance by sound class. We note that overall the two models perform comparably, with the CRNN performing notably better on air conditioner and gunshot events while the CNN performs better on car horns and jackhammers. By design, we generated URBAN-SED to contain soundscapes with a range of characteristics, and in particular different polyphonies, allowing us to easily break down model performance by maximum polyphony. Since the models perform comparably, for the breakdown we focus on the CNN model. In Figure 3 (top) we present the F-measure, Precision and Recall (1 s segment-based) yielded by the CNN model on the test set, grouped by the maximum polyphony which ranges from one to seven. As one might expect, we note that the F-measure gradually declines as the maximum polyphony increases, but more interesting still, we see that it is because the recall declines, whereas the precision remains stable (and even goes up). This suggests that as more sound events overlap the model is increasingly likely to only detect a subset of the events, however it remains equally precise. We shall revisit this result in the following section when we consider human annotation performance. Finally, Scaper also allows us to easily assess model performance as a function of sound event SNR. A priori this seems complicated since every soundscape contains multiple events with heterogeneous SNR values. Scaper offers an easy solution: we took the 2000 JAMS annotations of the test set, edited them such that all events in a soundscape have the same SNR, and then re-generated the soundscape audio files from the modified JAMS annotations. We repeated this process eight times, setting the event SNR to be in the range 6–9, 9–12, 12–15, 15–18, 18–24, 24–27 and 27–30. This results in eight versions of the test set which have identical characteristics with the exception of the SNR, allowing for a highly controlled experiment that would not be possible otherwise. The performance of the CNN as a function of SNR is presented in Figure 3 (bottom). We see an interesting effect: as the SNR increases, the model's precision and recall display oppo-

[3] http://urbansed.weebly.com/

[4] https://git.io/v9GEM

[5] Evaluating the CRNN at finer temporal resolutions (100 ms and 20 ms) did not result in higher overall or class-wise F-measures compared to 1 s.

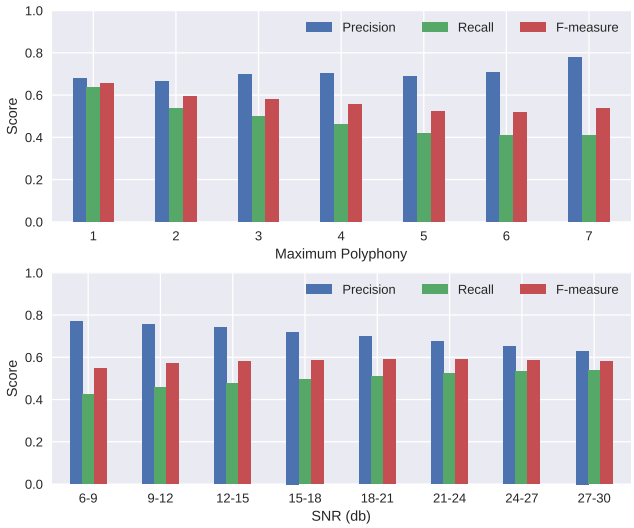

Figure 3: CNN score vs. max polyphony (top) and SNR (bottom).

site behaviors: recall goes up, meaning the model correctly detects more of the events, but this leads to an increased number of false positives, bringing down precision. The net effect is a relatively stable F-measure, but through this experiment we now know that this stability hides quite different model behavior as the SNR changes.

5. SCAPER FOR CROWDSOURCING EXPERIMENTS

As a second use case for Scaper, we briefly present a subset of the results from a recent study on crowdsourcing human annotations of sound events in soundscapes [31]. Crowdsourcing annotations can be seen as a complementary solution to synthesis: on the one hand it does not allow for the same level of control over the audio material, but on the other hand it supports the large-scale annotation of real audio recordings, which are necessary for obtaining a reliable estimate of how well a model will generalize once deployed in a real environment. Since the goal of the experiment was to assess the quality of human labels, it is not possible to use human labeled soundscapes as stimuli for the experiments. By using Scaper, we were able to generate audio stimuli with perfect annotations (a soundbank of 90 carefully curated sound events was used to generate 3000 soundscapes from which 60 were selected to cover a range of soundscape complexities and used as stimuli), thus ensuring that any effect observed in the experiment is due to the subjects' annotation abilities and the experimental interventions, and not due to a subjective discrepancy between the subjects and a "reference annotator". In Figure 4 we present the subjects' annotation performance in terms of segment-based precision, recall and F-measure computed at a 100 ms resolution as a function of maximum polyphony. In this experiment maximum polyphony was grouped into three levels, level 0 (maximum polyphony of 1), level 1 (maximum polyphony of 2) and level 2 (maximum polyphony of 3 or 4). Interestingly, we see that the human annotators exhibit similar behavior to the machine learning models: as the polyphony level increases the F-measure decreases, primarily due to a drop in recall, while the precision remains high. The human subjects, like the machine it seems, miss more events as the degree of overlap increases, but annotate the events that they do recognize accurately. This is a very promising result, suggesting that human labels for

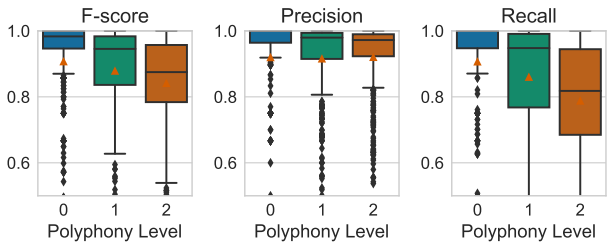

Figure 4: Human annotation performance vs. max polyphony: level 0 (max polyphony of 1), level 1 (2) and level 2 (3 or 4).

dense soundscapes can be considered reliable, albeit incomplete. Another goal of the experiment was to assess whether the way in which the audio was visualized in a web-based annotation interface had any influence on the quality of the annotations, where we compared three different visualizations: a waveform, a spectrogram, and no visualization at all. By comparing the distribution of human annotated event onsets and offsets to the Scaper annotations, we were able to show that a spectrogram visualization resulted in a statistically significant improvement in the temporal accuracy of the human annotations. For further details and results from the crowdsourcing experiments the reader is referred to [31].

6. DISCUSSION

In addition to the use cases presented above, Scaper could be used for soundscape augmentation, for example by adding sound events to an existing dataset, allowing the expansion of the set of classes a model is trained to recognize without requiring an entirely new dataset. Given a dataset for SED, one could also carve out all sound events that do not overlap with others, and use them as a soundbank to generate completely new soundscapes as a form of data augmentation for training a model. Even though the source material is not new, by applying audio transformations and generating previously unseen polyphonies the augmented data could potentially improve the generalizability of the model – this remains to be shown. Finally, it is important to note the limitations of our solution. First and foremost, the generated soundscapes, even if they sound quite realistic in some cases, cannot encompass the richness and complexity of real soundscapes. This means that while Scaper is useful for generating datasets both for training models and for comparing model performance as a function of controlled acoustic characteristics, it cannot be used as a replacement for manually annotated real-world recordings, if we wish to estimate how well a model will perform in a real environment. Furthermore, since the specification is fully up to the user, it is possible to generate soundscapes that are not plausible, and so the soundscape parameters need to be chosen conscientiously and as a function of the specific domain application. Currently Scaper does not support explicitly controlling certain scene characteristics such as maximum (or average) polyphony, and we plan to add this functionality in the future. Despite these limitations, we believe Scaper is a highly valuable tool for data generation, augmentation, and controlled evaluation, and we look forward to the research community's feedback and contributions.

7. ACKNOWLEDGMENT

The authors would like to thank Emre Çakir for his correspondence and assistance in accurately reproducing the CRNN model.

8. REFERENCES

[1] C. Mydlarz, J. Salamon, and J. P. Bello, "The implementation of low-cost urban acoustic monitoring devices," *Applied Acoustics*, vol. In Press, 2016.

[2] J. Salamon, J. P. Bello, A. Farnsworth, M. Robbins, S. Keen, H. Klinck, and S. Kelling, "Towards the automatic classification of avian flight calls for bioacoustic monitoring," *PLOS ONE*, vol. 11, no. 11, p. e0166866, 2016.

[3] Large-scale weakly supervised sound event detection for smart cars. [Online]. Available: http://www.cs.tut.fi/sgn/arg/dcase2017/challenge/task-large-scale-sound-event-detection

[4] R. Radhakrishnan, A. Divakaran, and P. Smaragdis, "Audio analysis for surveillance applications," in *IEEE Worksh. on Apps. of Signal Processing to Audio and Acoustics (WASPAA'05)*, New Paltz, NY, USA, Oct. 2005, pp. 158–161.

[5] S. Hershey et al., "CNN architectures for large-scale audio classification," in *IEEE ICASSP*, New Orleans, LA, USA, Mar. 2017, pp. 131–135.

[6] T.-W. Su, J.-Y. Liu, and Y.-H. Yang, "Weakly-supervised audio event detection using event-specific gaussian filters and fully convolutional networks," in *IEEE ICASSP*, New Orleans, LA, USA, Mar. 2017, pp. 791–795.

[7] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.

[8] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis." in *International Conference on Document Analysis and Recognition*, vol. 3, Edinburgh, Scottland, UK, Aug. 2003, pp. 958–962.

[9] B. McFee, E. Humphrey, and J. Bello, "A software framework for musical data augmentation," in *16th Int. Soc. for Music Info. Retrieval Conf.*, Malaga, Spain, Oct. 2015, pp. 248–254.

[10] D. Zotkin, R. Duraiswami, and L. Davi, "Rendering localized spatial audio in a virtual auditory space," *IEEE Trans. Multimedia*, vol. 6, no. 4, pp. 553–564, Aug. 2004.

[11] C. Verron, M. Aramaki, R. Kronland-Martinet, and G. Pallone, "A 3-d immersive synthesizer for environmental sounds," *IEEE TASLP*, vol. 18, no. 6, pp. 1550–1561, 2010.

[12] D. Schwarz, "State of the art in sound texture synthesis," in *Digital Audio Effects (DAFx)*, Paris, France, Sep. 2011, pp. 221–231.

[13] G. Lafay, M. Lagrange, M. Rossignol, E. Benetos, and A. Röbel, "A morphological model for simulating acoustic scenes and its application to sound event detection," *IEEE/ACM Trans. on Audio, Speech, and Lang. Proc.*, vol. 24, no. 10, pp. 1854–1864, Oct. 2016.

[14] E. J. Humphrey, J. Salamon, O. Nieto, J. Forsyth, R. Bittner, and J. P. Bello, "JAMS: A JSON annotated music specification for reproducible MIR research," in *15th Int. Soc. for Music Info. Retrieval Conf.*, Taipei, Taiwan, Oct. 2014, pp. 591–596.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=1953048.2078195

[16] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[17] F. Chollet, "keras," https://github.com/fchollet/keras, 2017.

[18] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. Zapata, and X. Serra, "ESSENTIA: an audio analysis library for music information retrieval," in *14th Int. Soc. for Music Info. Retrieval Conf.*, Curitiba, Brazil, Nov. 2013, pp. 493–498.

[19] B. McFee et al., "librosa 0.5.0," 2017. [Online]. Available: https://doi.org/10.5281/zenodo.293021

[20] V. Maffiolo, "Semantic and acoustic characterization of urban environmental sound quality," Ph.D. dissertation, Univ. du Maine, France, 1999.

[21] C. Guastavino, "The ideal urban soundscape: Investigating the sound quality of French cities," *Acta Acustica United with Acustica*, vol. 92, no. 6, pp. 945–951, 2006.

[22] J. H. McDermott, M. Schemitsch, and E. Simoncello, "Summary statistics in auditory perception," *Nature Neurosci.*, vol. 16, no. 4, pp. 493–498, 2013.

[23] R. Bittner, E. Humphrey, and J. Bello, "Pysox: Leveraging the audio signal processing power of sox in python," in *17th Int. Soc. for Music Info. Retrieval Conf., Late Breaking and Demo Papers*, New York City, NY, USA, Aug. 2016.

[24] E. M. Grimm, R. Van Everdingen, and M. J. L. C. Schöpping, "Toward a recommendation for a European standard of peak and LKFS loudness levels," *SMPTE Motion Imaging Journal*, vol. 119, no. 3, pp. 28–34, Apr. 2010.

[25] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *22nd ACM International Conference on Multimedia (ACM-MM'14)*, Orlando, FL, USA, Nov. 2014, pp. 1041–1044.

[26] E. Çakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," *IEEE/ACM TASLP, Special Issue on Sound Scene and Event Analysis*, In press, 2017.

[27] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, Mar. 2017.

[28] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *32nd Int. Conf. on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37, Lille, France, Jul. 2015, pp. 448–456.

[29] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd Int. Conf. on Learning Representations (ICLR)*, San Diego, CA, USA, May 2015.

[30] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," *Applied Sciences*, vol. 6, no. 6, p. 162, 2016.

[31] M. Cartwright, A. Seals, J. Salamon, A. Williams, S. Mikloska, D. MacConnell, E. Law, J. Bello, and O. Nov, "Seeing sound: Investigating the effects of visualizations and complexity on crowdsourced audio annotations," *Proceedings of the ACM on Human-Computer Interaction*, vol. 1, no. 1, 2017.