AC/DCIM: Acoustic Channels for Data Center Infrastructure Monitoring

Lars Wüstrich*, Sebastian Gallenmüller*, Marc-Oliver Pahl†, Georg Carle*
*Chair of Network Architectures and Services, Technical University of Munich, Garching near Munich, Germany
†Chair of Cybersecurity for Critical Networked Infrastructures, IMT Atlantique, Rennes, France

***Abstract*—Data center infrastructure monitoring (DCIM) uses various features to track a data center's state. In addition to collecting device-level information, the monitoring also includes physical features such as temperature or power intake to detect equipment failures and anomalies. Measuring physical features requires dedicated sensors at specific vantage points for efficient and reliable data collection. We propose a novel approach for DCIM using acoustic channels. Audio-based DCIM offers substantial benefits: sensors are affordable, data collection is non-intrusive, and audio processing is well-understood. Information extraction from acoustic channels can be challenging due to audio consisting of multiple devices' mixed and often noisy signals. Our paper demonstrates the feasibility of single-node state detection over audio side-channels. Experiments in a real data center show that our sound-based approach can successfully detect errors.**

***Index Terms*—data center monitoring, side-channels, anomaly detection**

I. Introduction

Data centers (DCs) house servers at scale to provide computing power and data storage capabilities to customers. The availability of DCs heavily influences our daily lives since they provide data for various applications. The criticality of their constant availability leads to the meticulous monitoring of DCs via data center infrastructure monitoring (DCIM) [1]. DCIM combines multiple indicators such as energy consumption, temperature, or humidity. Using these information channels, problems may be detected in advance to prevent failures and minimize service outages.

While some devices offer monitoring capabilities such as IPMI [2], not all hardware supports these standards. In addition, IPMI-like services may be disabled minimizing a machine's attack surface and avoid vulnerabilities introduced by IPMI itself [3].

We suggest using audio side-channels to gather additional data about a DC's current state. Audio-signal monitoring offers attractive benefits: collection is fast, well-understood, easy, affordable, and non-intrusive. It can detect device states without introducing a new attack vector to hosted devices.

The extraction of usable data is challenging. The soundscape of a DC contains the sound of all devices running, e.g., switches, servers, or air conditioning blended into a single information channel. We propose methods extracting valuable information from the soundscape of a DC and demonstrate their applicability to monitor specific events in the real world. We extract information about server behavior from a DC audio signal and detect device error codes in an audio stream.

978-1-6654-0601-7/22/$31.00

The remainder of the paper is structured as follows: We investigate related work in Section II before presenting background information in Section III. In Section IV, we present our approach to leverage sound for DCIM and evaluate our method in Section V. Section VI concludes the paper.

II. Related Work

The private sector mainly drives DCIM progress [1]. Table I summarizes commonly used DCIM features and related work based on them.

Depending on the used features, DCIM may require expensive hardware, e.g., power meters or heat image cameras. Typically, measurement hardware must be placed at critical locations, e.g., the power lines, to collect accurate information. In contrast, the still unused audio channels can be measured non-intrusively, flexibly, and inexpensively through microphones. While previous works utilize various side channels for DCIM, to the best of our knowledge, none use sound.

Levy et al. [1] present a holistic approach for DCIM via IoT sensors. They propose using IoT sensors to include physical properties in the monitoring, such as vibration in cabinets, differential air pressure, and water pressure of cooling systems.

Many approaches monitor power consumption at various levels to detect abnormal behavior. Dayarathna et al. [4] investigate methods to model the energy consumption of DCs. Their survey presents techniques to model power consumption on different levels of a DC. Borghesi [5] analyzes the power consumption of single nodes in HPC environments to detect anomalies. The presented prototype achieves an anomaly detection accuracy of 95%.

Marwah et al. [6] present an approach to predict equipment failure based on heat emission of DC devices. Similarly, Lee et al. [7], [8] use thermal measurements to detect anomalies in DCs. Ahuja [9] analyzes the importance of airflow management in DCs to optimize cooling capabilities. Rodriguez [10] uses a wireless sensor network to support its DCIM. In addition to temperature, deployed sensors monitor humidity to optimize cooling capabilities.

III. Data Center Infrastructure Monitoring

DCIM combines sophisticated monitoring and management tools to monitor large-scale information technology (IT) equipment hosting. These tools have the purpose of capturing the state of devices and enabling operators to make changes. Each device has a physical state and a closely coupled cyber state.

TABLE I
PHYSICAL FEATURES OF DCIM USED BY RELATED WORK

Feature	RW
Power consumption	[1], [4], [5]
Heat	[1], [6]–[8]
Airflow	[1], [9]
Humidity	[1], [10]
Vibration, water/air pressure	[1]
Sound	None

For example, a server performing complex computations will increase the clock of its CPUs. High computational load affects a device's power intake and thermal emission, influencing cooling. In the following sections, we describe the features of both, state types and the means to monitor them.

a) Data center devices: In addition to IT equipment like servers, routers, and switches, DCs host additional hardware for their operation. Power supply units, power distribution units, switchgear, electrical panels, and generators ensure constant power provisioning for hosted devices. DCs also contain complex cooling systems consisting of chillers, cooling towers, and air conditioning units to manage the climate. Finally, to automate its operation, DCs often use automation systems for managing their building infrastructure. To address emergencies, DCs also have fire systems in place. [1]

DCs have two layers: A *device layer* consisting of IT equipment for computational and data storage capabilities, and a *physical layer* for managing the environment for the device layer. To ensure an efficient and optimal operation of DCs, operators constantly monitor and manage both layers.

b) Device Level Monitoring and Management: To monitor and manage IT equipment, operators use tools to monitor and change the state of devices. While most tools rely on data collected within a host OS, there are additional tools that work without a running OS. Tools running on top of an OS can monitor all sorts of information on a node, e.g., processor load, memory consumption, etc. There are various tools to monitor the state of machines on the cyber level. Within a running OS, tools like check_mk [11] or Nagios [12] enable data collection of device state on individual hosts. They can collect information about the workload, used memory, and running processes. Operators also collect information about data flow between hosted nodes. Tools like Tivoli Netcool and OMNIbus, or Grafana [13] accumulate the collected data and present a holistic overview of a DC's state [14].

Operators also have the capabilities to manage devices without physical interaction. IPMI [2] or iDRAC [15] provide API-like access over the network. These enable operators to access devices and request information, even when they are shut down or have no running OS.

c) Physical Level Monitoring and Management: Physical level monitoring and management address the physical properties of DCIM. These include the management of power supply and climate control. Distributed sensors collect the required information throughout DCs. For power management, power meters in distribution devices measure the power intake. Hardware vendors also provide tools for remote management

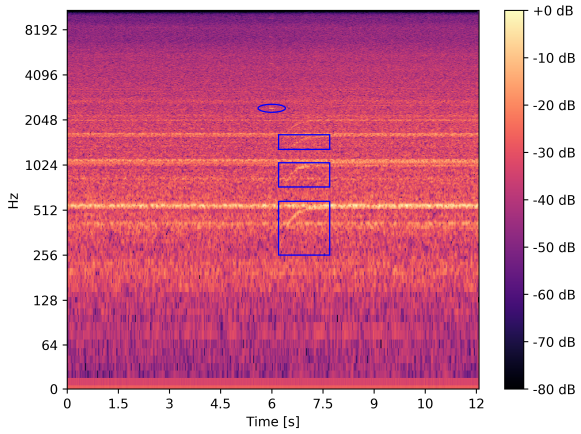

Fig. 1. Log power spectrogram of a recording in our DC. The ellipsis marks a beep that is emitted by a booting server. The rectangles mark the fans of a server spinning up during its startup.

of power supply. A DC's climate control ensures optimal room conditions for hosted hardware. To prevent devices from overheating, IT equipment has onboard cooling like fans. There are additional devices for climate control to handle the heat generated by the many devices in DCs. These are additional air conditioning units, cooling towers, and airflow management units between devices. Via distributed sensors located on devices and throughout infrastructure, DCIM collects information about temperature and humidity. It then uses this information to adjust and optimize the climate. Further, it is possible to enhance DCIM with a variety of small sensors to collect additional data, e.g., the vibration of a cabinet [1].

DCIM encompasses a variety of tools that collect information at several levels to reflect a DC's state.

IV. ACOUSTIC DCIM

To effectively use acoustic channels for DCIM, our method needs to extract and identify single device behavior from a mixed signal. Due to noisy background, single devices do not stick out in an analysis of a signal's amplitude and loudness.

To identify single device behavior, our method consists of two steps. First (IV-A), we manually perform spectral analysis to find relevant frequencies and limit the frequency band. Second (IV-B), our method automatically extracts the time frame during which an activity takes place.

A. Spectral Analysis

While most physical operations emit noise, sounds do not affect all frequencies of a spectrum. CPUs typically emit noise at high frequencies during processing [16], buzzers or fans emit sound on other frequencies.

To identify relevant frequencies, we perform spectral analysis. A power spectrogram shows the amount of energy carried by a frequency in a time frame. Figure 1 shows the full spectrogram of a recording from our DC. A visual inspection of the figure shows distinct frequencies constantly carrying high energy, while others carry low energy. Spinning fans emit noise on the high energy frequencies. Analyzing a spectrogram over a time frame enables the identification of energy changes

on specific frequencies. The recording in Figure 1 contains the startup of a server. At around 6 s a buzzer emits a short high-pitched beep at 2500 Hz. The traces starting at 6.5 s show the acceleration of the server's fans to their normal operating speed until 7.2 s. Since multiple identical servers run in the background, the traces end in significant frequencies. The ellipsis in Figure 1 marks the beep in the spectrogram; the squares mark the accelerating fans.

There are two types of device behavior: (1) single-frequency events and (2) multi-frequency events. After identifying the relevant frequencies on the spectrum, we apply a band-pass filter to remove non-relevant frequencies for the next step.

B. Time Frame Extraction

The second step identifies the time frame of events in a signal. Since uptime is a primary objective for DCs, devices are constantly running. State changes only occur due to failing equipment, scheduled events (e.g., maintenance), or unscheduled events (e.g., reboot due to software failures).

The design of DCs aims at minimum physical presence of humans. This results in a constant soundscape and spectrogram that only changes if devices alter their physical state.

In noisy DC environments, it is impossible to use spectral flatness (SF) [17] to determine the beginning or end of events. SF indicates how tone-alike a signal is by value in $[0, 1]$. The closer SF is to 1, the more tone-alike the signal is. Due to the high amount of background noise, DC recordings have a low SF. The SF throughout the recording in Figure 1 does not exceed 0.0005. The use of SF is only possible in quiet environments where monitored events start and end in silence, which is not the case for DCs.

Due to the constant background noise, we can detect spectrographic changes by calculating the *root mean square* (RMS) energy. The RMS energy indicates the total energy a sample carries across all frequencies. Changes in device behavior cause changes in the RMS energy. To identify trends in the noisy signal, we smooth the RMS energy curve using a moving average. While smoothing helps to identify general trends, it hides short-lived spikes or gaps. Therefore, smoothing depends on the type of the searched event and the number of RMS energy frequencies. The final step extracts the start and end of activities from the smoothed curve. Our approach automatically determines begin and end of an event through a threshold-based method. RMS energy exceeding a pre-defined threshold t_s in a given time window w_s, marks the beginning of an event. Analog, we detect the ending if the RMS energy falls below a threshold t_e in a window w_e.

This pre-processing can then help identify relevant subsequences from measurements for further processing. For example, image recognition on spectrograms can identify the type of an event [18].

V. Experiments

We conduct several measurements in a DC to evaluate the feasibility of our approach. Initially, we demonstrate the soundscape during normal operation, afterward, we apply our methods to failure detection. The air-conditioned DC consists of 45 servers and 7 actively cooled switches. We conducted our measurements during regular operation with normal background noise. For our measurements, we set up a Blue Yeti X [19] microphone and recorded the audio signal at a distance of approx. 1.5 m.

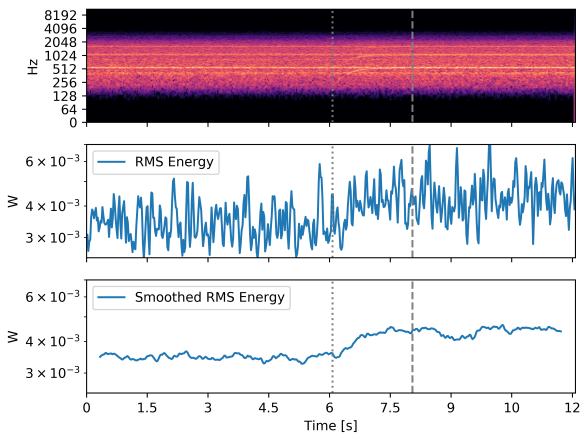

Fig. 2. A server spinning up its fans during boot. The top part shows the log power spectrogram of the filtered signal. The middle plot shows the RMS energy for the spectrogram. The bottom plot shows the smoothed RMS energy. The dotted and dashed line mark start and end of the identified event.

a) Normal operation: In our first experiment, we instructed single devices to execute commands that physically alter their behavior, e.g., power-off, power-on, or reboot. For each operation, we created an audio side-channel recording. To analyze recordings, we implemented the proposed approach with the librosa python library [20].

Figure 1 shows the spectrogram of our first experiment. The recording has a length of 12 s and contains a short, single-frequency event and a longer multi-frequency event. At the 6 s mark, a server performs a startup. After booting, the server remains turned on and is available to users.

We aim to identify the time frame during which the server accelerates its fans and preceding beep. The accelerating fans cause energy traces between 256 Hz and 512 Hz, 512 Hz and 1024 Hz, and 1300 Hz and 1600 Hz while the beep occurs at 2450 Hz. Since multiple devices of the same type are running in the background, the noise traces merge with significant frequencies in the background noise. To detect the accelerating fans, we limit the frequency band to 256 Hz–1700 Hz, which covers all frequencies affected by the fans. The upper part of Figure 2 shows the limited frequency band used for calculating the RMS energy. The middle part of the figure shows the calculated RMS energy of the filtered signal and the smoothed RMS energy in the lower part. Since we calculate the RMS energy for a wide frequency band, the smoothing considers 30 neighboring samples. The RMS energy remains constant, until it increases during the server's startup. The RMS energy stays constant after the fans reach operational speed. By using our method, we can automatically identify the period of the server's physical startup. The dotted and dashed lines in Figure 2 show the detected duration of the startup.

Our experiment implicates that the number of running

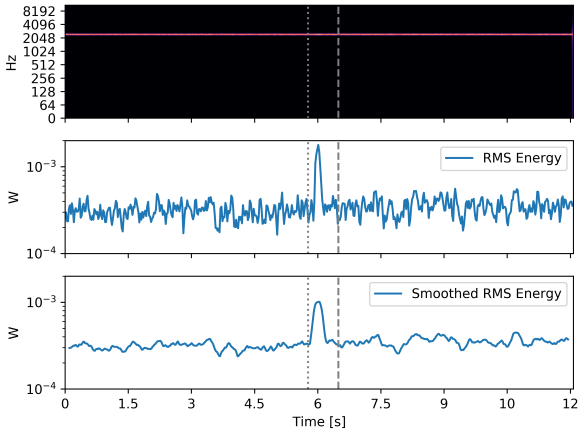

Fig. 3. Detection of a short single-frequency event in our first experiment

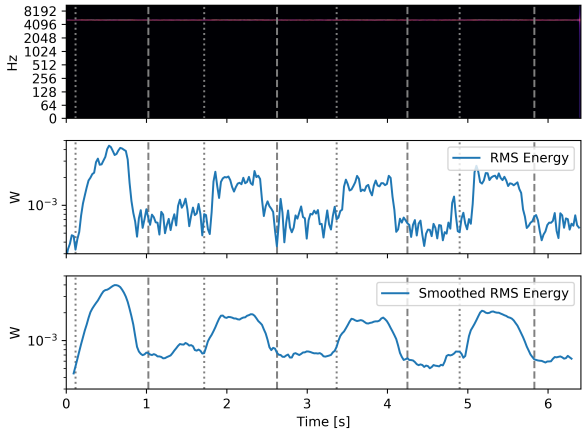

Fig. 4. Error code emitted by a server's buzzer due to a faulty fan

devices affects the total RMS energy. However, we have not investigated if it is possible to infer the number of running devices from the RMS yet. To identify short single-frequency events, we limit the frequency band to the relevant frequency, in case of the beep, 2400 Hz–2500 Hz, and reduce the smoothing. Since we consider fewer frequencies for single-frequency event detection, the calculated RMS energy is less noisy. Figure 3 visualizes this part of the experiment. We can detect this type of event through minor adjustments to the thresholds t_s and t_e and window sizes w_s and w_e.

b) Failure detection: After demonstrating acoustic channel analysis during normal operation, we apply our method to identify erroneous server states reflected by a beep code. We unplugged a server's fan such that it gets detected as faulty. As a result, the manipulated server emits a series of beeps to indicate the issue. Each beep has a length of roughly 1 s at 5150 Hz. To detect such short-time events we reduce the smoothing similar to the first experiment. Since the event occurs at 5150 Hz, we limit the frequency band to 5100 Hz–5200 Hz. Figure 4 shows the results of the second experiment. The lower plots show that the RMS energy spikes during the server's buzzer activity. Each beep gets automatically detected by our method (dotted and dashed lines). This shows that we could reliably extract the error code from the audio signal.

Our evaluation shows that acoustic side channels are applicable to detect events in DCIM, despite noisy background.

c) Discussion: We conducted two experiments addressing two important use cases to show the potential and relevancy of our approach for DCIM. A server's fan activity can give insights into its state and indicate potential issues. The early detection of a failing fan can help prevent further hardware damage, e.g., by overheating. The detection of fan activity requires monitoring a wide frequency band, which is potentially noise-prone. However, our experiments show that we can successfully identify such an event from a noisy signal without OS-level information or external tools like IPMI. Our second experiment revolves around the detection of beep codes emitted by buzzers. Devices can emit beeps when booting. However, they also use beeps to indicate the presence and type of errors, e.g., faulty RAM or failing fans. Since buzzers emit noise at distinct frequencies, we can limit the monitored spectrum to these frequencies to detect activity. Reducing the analyzed frequency band also reduces the background noise, making it easier to detect relevant events. Our experiments show that we can identify error codes purely by analyzing audio without additional tools on devices.

VI. Conclusion

This paper presented an approach to use acoustic channels for DCIM. Despite the challenging environment—analyzing a blended information channel and noisy signal—we successfully derived the behavior of single devices. Using sound for DCIM has several advantages: it is inexpensive and easy to set up thanks to widespread usage in consumer hardware. Sound can be recorded with high granularity and accuracy, and its measurement is non-intrusive. Our measurements have shown that we can identify the sounds emitted by a server during normal operation, such as fan noise or beeps. More importantly, we applied our methodology to detect failing equipment in a DC. Our automatic detection mechanism identified a server's beep error codes without additional tools. We demonstrated the feasibility of acoustic channel DCIM, motivating the cause to consider them as source for DCIM.

In the future, we will evaluate the robustness of our approach with regards to the number of devices, concurrent failures, and how this affects our approach's accuracy. We plan to extend and refine our approach to identify actions executed by DC devices. We will explore how this can improve error detection capabilities, e.g., the reliable detection of a board's beep code due to faulty RAM. Further, we will correlate acoustic information with management traffic to verify a single device's behavior and detect anomalies. Our approach can also be extended by using multiple microphones, enabling the localization of faulty devices. Finally, we will further explore how our approach complements existing DCIM solutions.

Acknowledgments

This work has been supported by the German Federal Ministry of Education and Research, project SKINET, grant 16KIS1221 and the German-French Academy for the Industry of the Future in TRUE-VIEW.

References

[1] M. Levy and J. O. Hallstrom, "A New Approach to Data Center Infrastructure Monitoring and Management (DCIMM)," in *IEEE 7th Annual Computing and Communication Workshop and Conference, CCWC 2017, Las Vegas, NV, USA, January 9-11, 2017*. IEEE, 2017, pp. 1–6. [Online]. Available: https://doi.org/10.1109/CCWC.2017.7868412

[2] *- IPMI - Intelligent Platform Management Interface Specification Second Generation*, Intel, Hewlett-Packard, NEC, Dell, 10 2013, rev. 1.1.

[3] A. Bonkoski, R. Bielawski, and J. A. Halderman, "Illuminating the Security Issues Surrounding Lights-Out Server Management," in *7th USENIX Workshop on Offensive Technologies, WOOT '13, Washington, D.C., USA, August 13, 2013*, J. Oberheide and W. K. Robertson, Eds. USENIX Association, 2013. [Online]. Available: https://www.usenix.org/conference/woot13/workshop-program/presentation/bonkoski

[4] M. Dayarathna, Y. Wen, and R. Fan, "Data Center Energy Consumption Modeling: A Survey," *IEEE Commun. Surv. Tutorials*, vol. 18, no. 1, pp. 732–794, 2016. [Online]. Available: https://doi.org/10.1109/COMST.2015.2481183

[5] A. Borghesi, A. Libri, L. Benini, and A. Bartolini, "Online Anomaly Detection in HPC Systems," in *IEEE International Conference on Artificial Intelligence Circuits and Systems, AICAS 2019, Hsinchu, Taiwan, March 18-20, 2019*. IEEE, 2019, pp. 229–233. [Online]. Available: https://doi.org/10.1109/AICAS.2019.8771527

[6] M. Marwah, R. Sharma, and C. Bash, "Thermal Anomaly Prediction in Data Centers," in *2010 12th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*. IEEE, 2010, pp. 1–7.

[7] E. K. Lee, H. Viswanathan, and D. Pompili, "Model-Based Thermal Anomaly Detection in Cloud Datacenters," in *IEEE International Conference on Distributed Computing in Sensor Systems, DCOSS 2013, Cambridge, MA, USA, May 20-23, 2013*. IEEE Computer Society, 2013, pp. 191–198. [Online]. Available: https://doi.org/10.1109/DCOSS.2013.8

[8] E. Lee, H. Viswanathan, and D. Pompili, "Model-Based Thermal Anomaly Detection in Cloud Datacenters Using Thermal Imaging," *IEEE Trans. Cloud Comput.*, vol. 6, no. 2, pp. 330–343, 2018. [Online]. Available: https://doi.org/10.1109/TCC.2015.2481423

[9] N. Ahuja, C. W. Rego, S. Ahuja, S. Zhou, and S. Shrivastava, "Real Time Monitoring and Availability of Server Airflow for Efficient Data Center Cooling," in *29th IEEE Semiconductor Thermal Measurement and Management Symposium*. IEEE, 2013, pp. 243–247.

[10] M. G. Rodriguez, L. E. O. Uriarte, Y. Jia, K. Yoshii, R. Ross, and P. H. Beckman, "Wireless Sensor Network for Data-Center Environmental Monitoring," in *2011 Fifth International Conference on Sensing Technology*. IEEE, 2011, pp. 533–537.

[11] checkmk - infrastructure and application monitoring. https://checkmk.com/. Accessed: 2022-01-12.

[12] W. Barth, *Nagios: System and Network Monitoring*. No Starch Press, 2008.

[13] Grafana: The open observability platform. https://grafana.com/. Accessed: 2022-01-12.

[14] A. Aimar, A. A. Corman, P. Andrade, S. Belov, J. D. Fernandez, B. G. Bear, M. Georgiou, E. Karavakis, L. Magnoni, R. R. Ballesteros *et al.*, "Unified Monitoring Architecture for IT and Grid Services," in *Journal of Physics: Conference Series*, vol. 898, no. 9. IOP Publishing, 2017, p. 092033.

[15] *- iDRAC - integrated Dell Remote Access Controller* , Dell.

[16] D. Genkin, A. Shamir, and E. Tromer, "Acoustic Cryptanalysis," *J. Cryptol.*, vol. 30, no. 2, pp. 392–443, 2017. [Online]. Available: https://doi.org/10.1007/s00145-015-9224-2

[17] N. Madhu, "Note on Measures for Spectral Flatness," *Electronics letters*, vol. 45, no. 23, pp. 1195–1196, 2009.

[18] V. Boddapati, A. Petef, J. Rasmusson, and L. Lundberg, "Classifying Environmental Sounds Using Image Recognition Networks," in *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 21st International Conference KES-2017, Marseille, France, 6-8 September 2017*, ser. Procedia Computer Science, C. Zanni-Merk, C. S. Frydman, C. Toro, Y. Hicks, R. J. Howlett, and L. C. Jain, Eds., vol. 112. Elsevier, 2017, pp. 2048–2056. [Online]. Available: https://doi.org/10.1016/j.procs.2017.08.250

[19] Blue - Yeti X. https://www.bluemic.com/en-us/products/yeti-x/. Accessed: 2022-01-12.

[20] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and Music Signal Analysis in Python," in *Proceedings of the 14th Python in Science Conference*, vol. 8. Citeseer, 2015, pp. 18–25.