# MODULATION SPECTRAL SIGNAL REPRESENTATION AND I-VECTORS FOR ANOMALOUS SOUND DETECTION

Technical Report

*Parth Tiwari*[1] [3]*, Yash Jain*[2]*, Anderson Avila*[3]*, João Monteiro*[3]*,*
*Shruti Kshirsagar*[3]*, Amr Gaballah*[3]*, Tiago H. Falk*[3]

[1] Department of Industrial and Systems Engineering, IIT Kharagpur, India
[2] Department of Mathematics, IIT Kharagpur, India
[3] MuSAE Lab, Institut National de la Recherche Scientifique - Centre EMT, Montreal, Canada

## ABSTRACT

This report summarizes our submission for Task-2 of the DCASE 2020 Challenge. We propose two different anomalous sound detection systems, one based on features extracted from a modulation spectral signal representation and the other based on i-vectors extracted from mel-band features. The first system uses a nearest neighbour graph to construct clusters which capture local variations in the training data. Anomalies are then identified based on their distance from the cluster centroids. The second system uses i-vectors extracted from mel-band spectra for training a Gaussian Mixture Model. Anomalies are then identified using their negative log likelihood. Both these methods show significant improvement over the DCASE Challenge baseline AUC scores, with an average improvement of 6% across all machines. An ensemble of the two systems is shown to further improve the average performance by 11% over the baseline.

***Index Terms*—** i-Vectors, Amplitude-Modulation Spectrums, Graph, Clustering, Gaussian Mixture Models

## 1. INTRODUCTION

Monitoring industrial machinery can prevent the production of faulty products and decrease the chances of machine breakdown. Anomalous sounds can indicate symptoms of unwanted activity, hence, Anomalous Sound Detection (ASD) systems can potentially be used for real time monitoring of machines. Unsupervised ASD systems rely on only "normal" sounds for identifying anomalies. Developing ASD systems in an unsupervised manner is essential, as: (i) the nature of anomalies may not be known beforehand, and (ii) deliberately destroying expensive devices is impractical from a development cost perspective. In addition, the frequency at which anomalies occur is low and the variability in the type of anomaly is high, therefore, developing balanced datasets for supervised learning is difficult.

In our proposed systems, we focus on using features which are able to capture anomalous behaviour. Simple machine learning algorithms when used on top of these features are able to beat the baseline performance[1]. In our first system, we propose an outlier detection method which is similar to a nearest neighbour search. In this method, clusters of normal sounds are formed using a nearest neighbour graph to capture variations in the normal working sounds of a machine. Anomalies are then identified based on their distance from these clusters. Modulation spectrum features are used for this system. The features are extracted from the so-called modulation spectrum (MS) signal representation, which quantifies the rate of change of the signal spectral components over time. These features have previously been useful for stress detection in speech [2], for speech enhancement [3], and room acoustic characterization [4], to name a few applications.

In our second system, in turn, we use i-vectors and Gaussian Mixture Models (GMM) for anomaly detection. i-vectors have been widely used for speech applications, including speech, speaker, language, and accent recognition. We extract i-vectors from MFCC features and use them to train GMMs for anomaly detection. The negative log likelihood for a sample is used as its anomaly scores. Lastly, an ensemble of these two systems is also experimented with.

## 2. SYSTEM DESCRIPTION

### 2.1. System 1 - Graph Clustering using Modulation Spectrograms

#### *2.1.1. Pre-processing and Feature Extraction*

Modulation spectrum corresponds to an auditory spectro- temporal representation that captures long-term dynamics of an audio signal. The pipeline proposed in [5] is used to extract modulation spectrograms.

Prior to feature extraction, noise reduction is performed on the signal through a spectral gating method (using *noisereduce*[1] in Python), described as follows: 100 normal training sound clips for a machine-id are averaged and used as a noise clip for that machine-id. An FFT is calculated over this noise clip and statistics including the mean power, are tabulated for each frequency band. A threshold for each frequency band is calculated based upon the statistics. An FFT is calculated over the signal. A mask is determined by comparing the signal FFT to the threshold. The mask is smoothed with a filter over frequency and time. The mask is appled to the FFT of the signal, and is inverted.

The speech activity level is normalized to -26 dBov (dB overload), after noise removal thus eliminating unwanted energy variations caused by different loudness levels in the speech signal. Next, the pre-processed speech signal $\hat{x}(n)$ is filtered by a 60-channel gammatone filterbank, simulating cochlear processing [6]. The first filter of the filterbank is centered at 125 Hz and the last one at at just below half of the sampling rate. Each filter bandwidth follows the

[1] `https://pypi.org/project/noisereduce/`

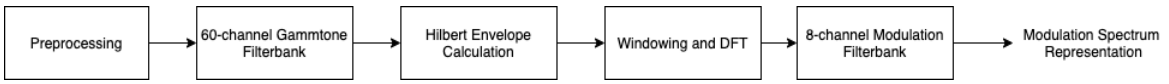


Figure 1: Block diagram describing steps for computing the modulation spectral representation

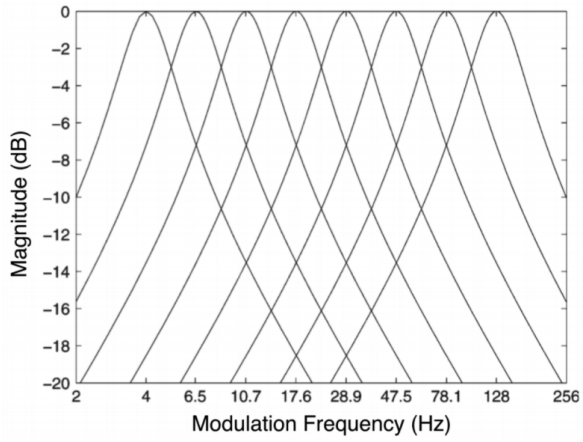


Figure 2: Frequency responses of the 8-channel modulation filterbank

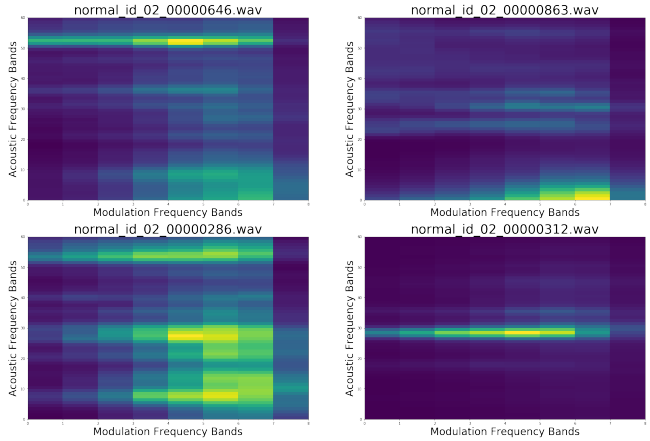


Figure 3: Modulation spectrograms for four normal training samples from Pump machine-id 2

equivalent rectangular bandwidth (ERB), which is an approximation of the bandwidths of the filters in human hearing, as described below:

$$ERB_{\mathrm{j}} = \frac{f_{\mathrm{i}}}{Q_{\mathrm{ear}}} + B_{\mathrm{min}}, \tag{1}$$

where $f_{\mathrm{j}}$ represents the center frequency of the j-th filter. $Q_{\mathrm{ear}}$ represents the asymptotic filter quality at high frequencies and $B_{\mathrm{min}}$ is the minimum bandwidth for low frequencies. They are set, respectively, to 9.265 and 24.7.

The temporal envelope $e_{\mathrm{j}}(n)$ is then computed from $\hat{x}_{\mathrm{j}}(n)$, the output of the j-th acoustic filter, via the Hilbert transform:

$$e_{\mathrm{j}}(n) = \sqrt{\hat{x}_{\mathrm{j}}(n)^2 + \mathscr{H}\{\hat{x}_{\mathrm{j}}(n)\}^2} \tag{2}$$

where $\mathscr{H}\{.\}$ denotes the Hilbert Transform. Temporal envelopes $e_{\mathrm{j}}(n), j = 1, ..., 60$ are then windowed with a 256-ms Hamming window and shifts of 40 ms. The discrete Fourier transform $\mathscr{F}\{.\}$ of the temporal envelope $e_{\mathrm{j}}(m; n)$ (m indexes the frame) is then computed in order to obtain the modulation spectrum $E_{\mathrm{j}}(m, f_{\mathrm{m}})$, i.e.,

$$E_{\mathrm{j}}(m; f_{\mathrm{m}}) = \|\mathscr{F}(e_{\mathrm{j}}(m; n))\| \tag{3}$$

where m represents the m-th frame obtained after every Hamming window multiplication and $f_{\mathrm{m}}$ designates modulation frequency. The time variable n is dropped for convenience. Lastly, following recent physiological evidence of a modulation filterbank structure in the human auditory system [6]. an auditory-inspired modulation filterbank is further used to group modulation frequencies into eight bands. These are denoted as $\mathscr{E}_{(\mathrm{j,k})}(m), k = 1, ..., 8$, where j indexes the gammatone filter and k the modulation filter. Figure 3 depicts the frequency response for the 8-channel modulation filterbank used in our system. Note that the filter center frequencies are equally spaced in the logarithmic scale from 4 to 128 Hz. The modulation spectral representation frames obtained over time are averaged for all our experminets. This results in a $60 \times 8$ modulation spectral representation (i.e., modulation spectrogram) for each sound clip.

### 2.1.2. *Anomaly Detection*

Figure 3 shows four modulation spectrograms for the normal training samples of Pump machine-id 2. It can be seen that a significant amount of variability exists within the same machine-id. We capture this variability by a graph-based clustering approach using modulation spectra as features.

Consider a graph $G = (V, E)$ where $V$ is the set of nodes comprising of the normal training sound clips. $E$ is the set of edges connecting the nodes. Two nodes $v_{\mathrm{p}}, v_{\mathrm{q}} \in V$ share an edge $e_{\mathrm{pq}} \in E$ such that $q = argmin(\{ D(v_{\mathrm{p}}, v_{\mathrm{r}}) \mid r = 1, .., p-1, p+1, ...|V| \})$. Here $D(v_{\mathrm{p}}, v_{\mathrm{r}})$ is the L1 distance between $v_{\mathrm{p}}, v_{\mathrm{r}}$. The graph $G$, when constructed in this manner, consists of several disjoint subgraphs i.e. $G = g_1 \cup g_2 \cup g_3 ... g_{\mathrm{n}}$ such that $g_{\mathrm{l}} \cap g_{\mathrm{m}} = \phi \ \forall \ l, m = 1, 2, ...n \mid l \neq m$. Each of these subgraphs is treated as a separate cluster. A centroid $\mu_{\mathrm{l}}$ and standard deviation $\sigma_{\mathrm{l}}$ are calculated corresponding to each cluster by taking the mean and standard deviation of all frequency bins. $\mu_{\mathrm{l}}, \sigma_{\mathrm{l}}$ both have dimensions $60 \times 8$. This graph $G$ and the corresponding cluster centroids, standard deviations are computed separately for each machine-id.

Anomaly score for a sound clip of a machine-id is calculated using the standard-deviation normalized distance from each cluster centroid corresponding to that machine-id. For a given sample in the test dataset $v_{\mathrm{t}} \notin V$, its anomaly score $A_{\mathrm{t}}$ is given by $A_{\mathrm{t}} = min(\{z_{\mathrm{t,l}} \mid \forall \ l\})$

$$z_{\mathrm{t,l}} = \sum_{j=1}^{60} \sum_{k=1}^{8} \frac{|z_{\mathrm{t,j,k}} - \mu_{\mathrm{l,j,k}}|}{\sigma_{\mathrm{t,j,k}}}. \tag{4}$$

Here, $z_{\mathrm{t,j,k}}$ is the energy value at the j$^{\mathrm{th}}$ gammatone filterbank

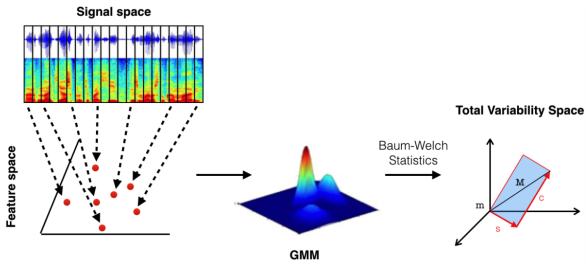


Figure 4: Diagram describing the steps for i-vector extraction

and k[th] modulation filterbank. The intuition behind this strategy for finding anomaly scores is that normal samples will lie closer the one of the cluster centroids in comparison to anomalous samples. This method does not require any training and can be seen as a KNN based ASD system where instead of computing distances from each training sample, we only find the distances from the cluster centroid.

### 2.2. System 2 - GMMs using MFCC i-Vectors

#### 2.2.1. *Feature Extraction*

The i-vector framework maps a list of feature vectors, $O = \{o_t\}_{t=1}^{N}$, where $o_t \in \mathbb{R}^F$, and $N$ is the frame index. Typically Mel-frequency cepstral coefficients (MFCC's) extracted from an utterance, into a fixed-length vector, $n \in \mathbb{R}^D$. In order to achieve that, a Gaussian mixture model (GMM), $\lambda = (\{w_k\}, \{m_k\}, \{\sigma_k\})$, is used. The GMM, trained on multiple utterances, is referred to as the universal background model (UBM), and is used to collect Baum-Welch statistics from each utterance [7]. Such statistics are computed for each mixture component $k$, resulting in the so-called supervector $M \in \mathbb{R}^{FK}$, where $F$ represents the feature dimension and $K$ is the number of Gaussian components. As in the Joint Factor Analysis (JFA) [8], the i-vector framework also considers that speaker and channel variability lies in a lower subspace of the GMM supervectors [9]. The main difference between the two approaches is that the i-vector projects both speaker and channel variability into the same subspace, namely total variability space, represented as follows:

$$M = m + Tw, \tag{5}$$

where $M$ is the dependent supervector (extracted from a specific utterance) and $m$ is the independent supervector (extracted from the UBM), $T$ corresponds to a rectangular low-rank total variability matrix and $w$ is a random vector with a normal distribution, the so-called i-vector. In our experiments, a 100-dimensional i-vector was adopted extracted on top of MFCC features.

*Mel frequency spectrum coefficients* - Prior to their extraction, the input signals (sampled at 16 kHz) are normalized to -26 dBOV. The signals also undergo a pre-emphasis filter of coefficient 0.95, which is meant to balance low and high frequency magnitudes. A 30-ms Hamming window with 50% overlap is applied before extracting the MFCCs. The Hamming window is used to remove edge effects [10]. The cepstral feature vector can then be extracted from each frame according to:

$$c_n = \sum_{m=1}^{M} [Y_m] cos\left[\frac{\pi n}{M}\left(\ m - \tfrac{1}{2}\ \right)\right], n = 1, 2, 3, ..., N, \tag{6}$$

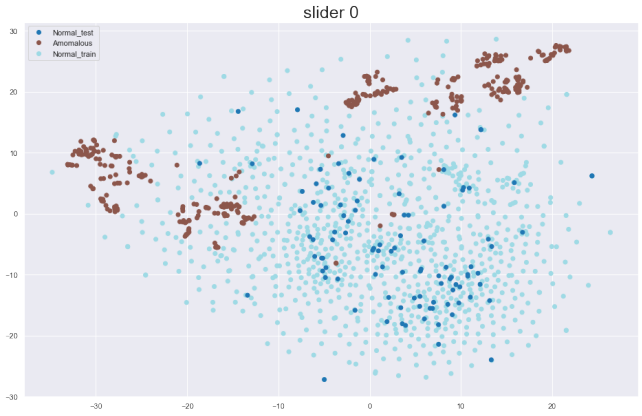


Figure 5: 2D t-SNE projections of i-Vectors corresponding to machine-slider, machine-id 0

where $c_n$ is the $n^{th}$ mel-cepstral coefficient and $Y_m$ refers to the log-energy of the $m^{th}$ filter. In this work, a set of 13 coefficients together with log energy, delta and delta-delta coefficients form the feature vector from each frame.

#### 2.2.2. *Anomaly Detection*

The 100 dimensional i-vectors extracted from the normal training sounds are used to train a Gaussian Mixture Model using scikit-learn [11]. Ten mixture components are used for all machines with each component having its own general full covariance matrix. The ability of i-vectors to capture anomalous behaviour is depicted in Figure 5. Here, 2D t-SNE embeddings show that i-vectors corresponding to anomalous sounds are well separated from the normal sounds. i-vectors seem to be spread according to a Gaussian density, which is the key motivation behind using GMMs for anomaly detection.

The negative log-likelihood of sample $x$ is given by:

$$-logP(x|\pi, \mu, \sigma) = -log\Big\{\sum_{k=1}^{K} \pi_k \mathbf{N}(x|\mu_k, \sigma_k)\Big\}, \tag{7}$$

where $\pi_k$ is the mixing coefficient for the $k^{th}$ component of the GMM and $\mu_k, \sigma_k$ are the corresponding mean and co-variance matrices. These values are used as the anomaly score.

### 2.3. System 3 - Graph–i-vector Ensemble

Lastly, an ensemble of the two proposed system shows performance improvement in several cases. Anomaly scores obtained from the two systems are first normalized using their minimum and maximum values. The ensemble anomaly scores are then computed by taking the geometric mean of the normalized values.

## 3. RESULTS

The results are shown in Table 1. When trained on development data[12, 13], the graph clustering system outperforms the baseline[1] by an average of 6% and 32% AUC for machines slider and valve respectively. The i-vector GMM system outperforms baseline and graph clustering system for some of the machine IDs

Table 1: AUC and pAUC scores for all machines in the development dataset. Modspec Graph, iVGmm, Ensemble correspond to Systems 1-3, respectively. The best scores for each case have been shown in bold.

| Machine | Mid | Baseline AUC | Modspec Graph AUC | iVGmm AUC | Ensemble AUC | Baseline pAUC | Modspec Graph pAUC | iVGmm pAUC | Ensemble pAUC |
|---|---|---|---|---|---|---|---|---|---|
| ToyCar | 1 | 81.36% | 78.24% | 75.04% | **81.64%** | **68.40%** | 64.69% | 57.54% | 66.75% |
| | 2 | 85.97% | 89.06% | 83.30% | **91.72%** | 77.72% | 76.14% | 67.00% | **79.78%** |
| | 3 | 63.30% | 67.16% | 79.47% | **78.21%** | 55.21% | 52.58% | 59.52% | **56.37%** |
| | 4 | 84.45% | 89.40% | 94.84% | **96.44%** | 68.97% | 63.54% | 82.94% | **84.80%** |
| | Avg | 78.77% | 80.96% | 83.16% | **87.00%** | 67.58% | 64.24% | 66.75% | **71.92%** |
| ToyConveyor | 1 | **78.07%** | 62.56% | 55.51% | 64.62% | **64.25%** | 51.59% | 52.82% | 52.24% |
| | 2 | **64.16%** | 54.03% | 53.80% | 56.65% | **56.01%** | 49.99% | 50.95% | 50.23% |
| | 3 | **75.35%** | 59.10% | 59.09% | 64.06% | **61.03%** | 50.31% | 52.82% | 52.25% |
| | Avg | **72.53%** | 58.57% | 56.13% | 61.78% | **60.43%** | 50.63% | 52.20% | 51.58% |
| fan | 0 | 54.41% | 63.37% | **67.85%** | 67.12% | 49.37% | 49.73% | **57.38%** | 52.92% |
| | 2 | 73.40% | 79.32% | 70.39% | **80.48%** | 54.81% | 57.16% | **61.93%** | 59.21% |
| | 4 | 61.61% | 71.76% | 73.52% | **78.07%** | 53.26% | 50.68% | **57.53%** | 53.99% |
| | 6 | 73.92% | 74.00% | 81.15% | **81.90%** | 52.35% | 49.38% | **56.31%** | 49.23% |
| | Avg | 65.83% | 72.11% | 73.23% | **76.89%** | 52.45% | 51.74% | **58.29%** | 53.84% |
| pump | 0 | 67.15% | 86.66% | 74.99% | **86.95%** | 56.74% | **82.52%** | 67.10% | 78.32% |
| | 2 | 61.53% | 62.44% | **74.91%** | 70.06% | 58.10% | 64.77% | 60.08% | **65.72%** |
| | 4 | 88.33% | 84.16% | **92.02%** | 90.73% | 67.10% | 59.95% | 73.74% | **68.00%** |
| | 6 | 74.55% | 81.64% | 71.10% | **82.65%** | 58.02% | 66.20% | 51.70% | **66.56%** |
| | Avg | 72.89% | 78.72% | 78.26% | **82.60%** | 59.99% | 68.36% | 63.15% | **69.65%** |
| slider | 0 | 96.19% | **99.91%** | 83.92% | 98.72% | 81.44% | **99.53%** | 50.04% | 93.44% |
| | 2 | 78.97% | **84.36%** | 56.93% | 77.93% | 63.68% | **73.86%** | 47.84% | 52.89% |
| | 4 | 94.30% | **97.83%** | 87.84% | 95.93% | 71.98% | **88.59%** | 62.71% | 79.69% |
| | 6 | 69.59% | **79.03%** | 59.04% | 71.40% | 49.02% | **55.47%** | 49.91% | 52.28% |
| | Avg | 84.76% | **90.28%** | 71.93% | 86.00% | 66.53% | **79.36%** | 52.63% | 69.57% |
| valve | 0 | 68.76% | **100.00%** | 79.33% | 98.80% | 51.70% | **100.00%** | 52.94% | 95.62% |
| | 2 | 68.18% | **99.88%** | 85.35% | 98.69% | 51.83% | **99.34%** | 56.27% | 93.29% |
| | 4 | 74.30% | **98.26%** | 84.10% | 95.88% | 51.97% | **91.32%** | 56.32% | 80.26% |
| | 6 | 53.90% | **89.22%** | 69.84% | 85.01% | 48.43% | **72.59%** | 49.91% | 59.65% |
| | Avg | 66.28% | **96.84%** | 79.65% | 94.59% | 50.98% | **90.81%** | 53.86% | 82.21% |

in pump and fan. The ensemble of graph clustering system and i-vector GMM system outperforms the baseline by an average AUC score of 8%,11% and 10% for machines ToyCar, fan, and pump, respectively. Interestingly, the performance for the ToyConveyor case was lower than that achieved by the benchmark system for all three proposed systems. This may be due to anomalies which occur for a very small time interval and are not being captured by the proposed longer-term features. We provide our implementation here[2].

## 4. REFERENCES


[1] Y. Koizumi, Y. Kawaguchi, K. Imoto, T. Nakamura, Y. Nikaido, R. Tanabe, H. Purohit, K. Suefusa, T. Endo, M. Yasuda, and N. Harada, "Description and discussion on DCASE2020 challenge task2: Unsupervised anomalous sound detection for machine condition monitoring," in *arXiv e-prints: 2006.05822*, June 2020, pp. 1–4. [Online]. Available: https://arxiv.org/abs/2006.05822

[2] A. R. Avila, S. R. Kshirsagar, A. Tiwari, D. Lafond, D. O'Shaughnessy, and T. H. Falk, "Speech-based stress classification based on modulation spectral features and convolutional neural networks," in *2019 27th European Signal Processing Conference (EUSIPCO)*. IEEE, 2019, pp. 1–5.

[3] S. Karimian-Azari and T. H. Falk, "Modulation spectrum based beamforming for speech enhancement," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2017, pp. 91–95.

[4] T. H. Falk and W.-Y. Chan, "Temporal dynamics for blind measurement of room acoustical parameters," *IEEE Transactions on Instrumentation and Measurement*, vol. 59, no. 4, pp. 978–989, 2010.

[5] T. H. Falk and W. Chan, "Modulation spectral features for robust far-field speaker identification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 1, pp. 90–100, 2010.

[6] M. Slaney, "An efficient implementation of the patterson-holdsworth auditory filter bank," 1997.

[7] D. Garcia-Romero and C. Y. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems," in *Twelfth Annual Conference of the International Speech Communication Association*, 2011.


[2] https://github.com/parth2170/DCASE2020-Task2

[8] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, "Joint factor analysis versus eigenchannels in speaker recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 4, pp. 1435–1447, 2007.

[9] J. H. L. Hansen and T. Hasan, "Speaker recognition by machines and humans: A tutorial review," *IEEE Signal processing magazine*, vol. 32, no. 6, pp. 74–99, 2015.

[10] B. Logan and et al., "Mel frequency cepstral coefficients for music modeling." in *Ismir*, vol. 270, 2000, pp. 1–11.

[11] 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," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[12] Y. Koizumi, S. Saito, H. Uematsu, N. Harada, and K. Imoto, "ToyADMOS: A dataset of miniature-machine operating sounds for anomalous sound detection," in *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, November 2019, pp. 308–312. [Online]. Available: https://ieeexplore.ieee.org/document/8937164

[13] H. Purohit, R. Tanabe, T. Ichige, T. Endo, Y. Nikaido, K. Suefusa, and Y. Kawaguchi, "MIMII Dataset: Sound dataset for malfunctioning industrial machine investigation and inspection," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, November 2019, pp. 209–213. [Online]. Available: http://dcase.community/documents/workshop2019/proceedings/DCASE2019Workshop_Purohit_21.pdf