At the Speed of Sound: Efficient Audio Scene Classification

Bo Dong[*§], Cristian Lumezanu[*], Yuncong Chen[*], Dongjin Song[*], Takehiko Mizoguchi[*]
Haifeng Chen[*], Latifur Khan[§]
NEC Laboratories America[*] University of Texas at Dallas[§]

ABSTRACT

Efficient audio scene classification is essential for smart sensing platforms such as robots, medical monitoring, surveillance, or autonomous vehicles. We propose a retrieval-based scene classification architecture that combines recurrent neural networks and attention to compute embeddings for short audio segments. We train our framework using a custom audio loss function that captures both the relevance of audio segments within a scene and that of sound events within a segment. Using experiments on real audio scenes, we show that we can discriminate audio scenes with high accuracy after listening in for less than a second. This preserves 93% of the detection accuracy obtained after hearing the entire scene.

ACM Reference Format:
Bo Dong[*§], Cristian Lumezanu[*], Yuncong Chen[*], Dongjin Song[*], Takehiko Mizoguchi[*] and Haifeng Chen[*], Latifur Khan[§]. 2020. At the Speed of Sound: Efficient Audio Scene Classification. In *2020 International Conference on Multimedia Retrieval (ICMR'20), June 8–11, 2020, Dublin, Ireland*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3372278.3390730

1 INTRODUCTION

Smart sensing platforms continually monitor their environment through cameras, microphones, or dedicated sensors (*e.g.*, temperature, humidity). To ensure safety and security, they must detect and react to ambient changes, such as new objects in the video frame, new sound events, or unusual sensor values [6]. While most smart sensing platforms primarily analyze video and sensor time series [23], recent studies demonstrate the benefit of *audio classification* not only as a complement but also as a replacement for video classification [1, 8].

We study audio scene classification (ASC)—the task of identifying the category of the surrounding environment using acoustic signals. Traditional audio scene classification methods train machine learning classifiers (*e.g.*, Gaussian Mixture Models, SVM) on *low-level* audio features (*e.g.*, MFCCs [24] or filter banks [10]), extracted with signal processing techniques [3]. More recently, deep learning architectures encode *high-level* features (or embeddings) that discriminate audio scenes [7, 14, 15].

Although they perform well, existing learning-based approaches require long audio sequences (10 to 30s) to be accurate; their performance drops as the duration of a scene decreases [15]. In many real life scenarios, there are not sufficient resources or time to "process" the acoustic environment thoroughly and it is critical to classify the category of a scene or audio event as efficiently as possible [1]. For example, self-driving cars must react quickly to the sound of police sirens, even when the police car is out of sight [21]; medical monitoring devices must raise alerts when they hear suspicious breathing sounds [8]; low battery IoT devices must be able to detect acoustic scenes locally without consuming too much energy [4].

To achieve efficient classification, our insight is to formulate ASC as a retrieval problem. We split the audio data into short segments (of less than a second), learn embeddings for each segment, and use the embeddings to classify each segment as soon as we "hear" it. Given a query segment (*e.g.*, short sound from the environment), we classify it in the class of the most similar historical segment, according to an embedding similarity function, such as the Euclidean distance. If we can accurately identify an audio scene after processing only a few such short audio segments, then we can issue faster alerts and save system resources by forgoing the processing of subsequent segments.

A natural question is how we can find embeddings that enable *accurate retrieval of short audio segments*. First, good embeddings must preserve *similarity*: segments part of the same audio scene should have similar embeddings. Second, they must capture the *importance* of each segment within a scene. For example, in a playground scene, the segments containing children laughter are more relevant for the scene; in contrast, silence or noise segments are less important since they can be found in many other types of scenes.

We present a deep learning framework to accurately classify audio environment after listening for less than a second. Our framework relies on recurrent neural networks and attention to learn embeddings for each audio segment. Key to the learning process is an optimization mechanism that minimizes an *audio loss function*. We construct this function to encourage embeddings to preserve segment similarity (through a distance-based component) and penalize nondescript segments while capturing the importance of the more relevant ones (with an importance-based component).

We use real world audio scenes [11] to demonstrate that listening to the *first 300ms* of a 10s audio scene is sufficient to detect its category (out of 15 possibilities) with high accuracy (around 70%). Our framework loses only around 7% of the accuracy obtained when listening to the entire scene. Furthermore, the classification accuracy increases to over 95% when the number of possible categories decreases.

Our contributions towards efficient audio scene classification are two fold. First, we break down the audio into short duration segments and classify them using nearest neighbor based retrieval. Then, to make retrieval faster, we construct a deep learning architecture to encode the audio segments by combining LSTM and attention. We train the encoder by optimizing a custom built audio

ICMR '20, June 8–11, 2020, Dublin, Ireland

ACM ISBN 978-1-4503-7087-5/20/06…$15.00
https://doi.org/10.1145/3372278.3390730

loss function that captures the importance of each segment in a scene and of each time step in a segment.

2 RELATED WORK

The first step in any audio processing framework is the extraction of low-level features. Popular features are mel frequency cepstral coefficients (MFCCs) [24], filter banks [10], mel scale spectrograms [9], or Gammatone spectrograms [16]. To augment these features, many frameworks add first order derivatives, eliminate background noise [12], or separate the harmonic and percussive parts [13]. Research on how to extract features from the raw audio is orthogonal to our work; we leave its exploration for the future.

Deep audio encoders feed low-level audio features into deep learning architectures to learn embeddings for fixed length scenes. Guo *et al.* [7] combine LSTM [18] and attention [20], while Phan *et al.* [14] propose an architecture composed of convolutional, recurrent, and attention layers (temporal and spatial). They train the encoder jointly with a classifier by minimizing the cross-entropy loss.

We also use attention and LSTM, but train the encoder separately by minimizing a custom audio loss; for detection, we use a nearest neighbor classifier. This allows us to classify a scene faster, *i.e.*, by computing embeddings for many short duration segments and retrieving the category of only the first (few) segments, without listening to the entire scene.

Turpault *et al.* also separate the training of the encoder and classifier [19]. They propose segment sampling strategies for triplet loss optimization in semi-supervised scene classification. We focus on building an improved loss function that can help emphasize the important audio segments in a fully supervised scenario. Although the goal of the audio loss is similar to that of attention mechanisms [5], note the subtle difference: attention reveals relevant time steps in a segment, audio loss relevant segments in a scene.

3 DESIGN

We describe the three components of our audio scene classification: raw audio processing, encoder, and loss optimization. Figure 1 presents the architecture. Our contributions lie in the encoder and loss optimization.

3.1 Raw audio processing

We decompose each audio scene using windowed FFT and extract 20 mel frequency cepstral coefficients [24]. We add their first derivatives and 12 harmonic and percussive features, known to enhance the raw feature set [13], to obtain 52 basic audio features for each FFT window.

Let $\mathbf{X} = (\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^n)^\mathrm{T} \in \mathbb{R}^{n \times T}$ represent an audio segment of length T (*e.g.*, of T consecutive FFT windows) with n basic features (where n=52). We associate each segment with the label of the scene to which it belongs. Our goal is audio segment retrieval: given a query segment, find the most similar historical segments using a similarity function, such as the Euclidean distance. We then classify our query segment in the same category as the most similar historical segment.

3.2 Learning audio embeddings

To perform efficient retrieval, we learn embeddings for each audio segment and compare the embeddings rather than the low-level audio features. We assume that the embedding is given by the following mapping function:

$$\mathbf{h} = \mathcal{F}(\mathbf{X}) \tag{1}$$

where $\mathbf{X} \in \mathbb{R}^{n \times T}$ is an audio segment of n basic features over T time steps and $\mathbf{h} \in \mathbb{R}^d$ is an embedding vector of size d. $\mathcal{F}$ is a non-linear mapping function.

We construct our mapping function to satisfy two criteria. First, it must encourage embeddings to reflect class membership. In other words, segments part of the same class should have similar embeddings, segments part of different classes different embeddings. Second, it must identify the important segments quickly, to enable efficient retrieval. We want to emphasize the segments that can discriminate a scene (*e.g.*, children laughter in a playground scene) and underplay those that are less descriptive (*e.g.*, noise).

We use a combination of bidirectional LSTM [18] and attention [20] to compute $\mathcal{F}$. We use LSTM to capture long-term temporal dependencies and attention to emphasize the important audio parts in a segment. As others have also shown [7, 14], recurrent networks and attention mechanism are efficient in identifying important features in audio.

We capture correlations between audio at different time steps in a segment by feeding all LSTM hidden states into the attention layer. Given a segment $\mathbf{X}$, denote by s_t the LSTM hidden state for timestep t. The attention weights encode the importance of each time step using a non-linear function $attn(s_t) = tanh(s_t\mathbf{V} + \mathbf{b})$ [20]. We normalize the attention weights using *softmax*:

$$a_t = \frac{\exp(attn(s_t))}{\sum_{t=1}^{T} \exp(attn(s_t))} \tag{2}$$

and compute the *similarity* embedding of the segment as the weighted average of each hidden state:

$$\mathbf{h}' = \sum_{t=1}^{T} a_t h_t \tag{3}$$

To capture the importance of each segment separately, we project the *similarity* embedding using a fully connected layer and compute the *importance* embedding:

$$\mathbf{h}'' = \mathbf{h}'\mathbf{V}' + \mathbf{b}' \tag{4}$$

We motivate the choice of *importance* embedding below when we discuss the loss function. The final embedding of a segment is a concatenation of $\mathbf{h}'$ and $\mathbf{h}''$:

$$\mathbf{h} = [\mathbf{h}'; \mathbf{h}''] \tag{5}$$

We learn all weights, including $\mathbf{V}$, $\mathbf{V}'$, and the LSTM weights, jointly when training the mapping $\mathcal{F}$. The *similarity* and *importance* embeddings are complementary in highlighting the discriminative segments in an audio scene. The *similarity* embedding helps identify the useful time steps *within a segment*, while the *importance* embedding the relevant segments *within a scene*.

3.3 Audio Loss

We train our network by optimizing a custom audio loss function. We construct our loss function to optimize for the two criteria used to construct the mapping $\mathcal{F}$: embeddings reflect class membership and emphasize important features within a class. To achieve

the first criteria, we use a distance-based component, such as the triplet loss [17]. Turpault *et al.* showed that triplet loss is effective in computing audio embeddings in semi-supervised audio scene detection [19].

For a triplet of audio segments a, p and n sampled from the training set, such that a and p have the same label and a and n have different labels (a stands for *anchor*, p for *positive*, n for *negative*), the loss is:

$$\mathcal{L}^{\text{sim}}_{a,p,n} = \max\left(\|\mathcal{F}(a) - \mathcal{F}(p)\|^2 - \|\mathcal{F}(a) - \mathcal{F}(n)\|^2, 0\right) \quad (6)$$

To emphasize segments with discriminative features, we augment the similarity loss to rely on the *importance* embeddings computed by Equation 4. This has the effect of penalizing the less important segments and training the network using the segments whose features are more discriminative. The final audio loss is:

$$\mathcal{L} = \sum_{\{a,p,n\}} (\mathbf{h}''_a \mathbf{h}''_p \mathbf{h}''_n)\mathcal{L}^{\text{sim}}_{a,p,n} - \alpha(\mathbf{h}''_a + \mathbf{h}''_p + \mathbf{h}''_n) \quad (7)$$

where $\mathbf{h}''$ represents the *importance* embeddings. α is a regularization parameter with values between 0 and 1. The first term ensures that only triplets where the three segments are simultaneously important are used in the similarity loss computation. The second term discourages the trivial solution that all three weights are zero.

To better understand how the loss optimization works, consider a few examples. When we select a triplet where all segments are important in discriminating their class, the second term of Equation 7 leads to an increase in the values of their *importance* embeddings. The effect of this triplet on the weights will be enhanced due to the product from the first term of the equation. On the other hand, when one or more segments are nondescript and do not affect the embedding, their *importance* embedding will be pushed towards 0. In essence, we use the *importance* embedding to filter the audio segments that we use in training the network.

Our encoder architecture is reminiscent of the neural machine translation [2] in that it combines LSTM and attention. However, unlike [2], we do not have a decoder and compute self attention between the encoder hidden states rather than attention between the decoder current state and the encoder hidden states. Unlike other audio encoders using LSTM and attention [7, 14], we explicitly encode the importance of a segment within its class into the segment's embedding and use the embedding to identify the important segments during the training process.

4 EVALUATION

In this section, we evaluate the accuracy of our audio scene classification framework using real world audio scenes.

4.1 Data and methodology

We evaluate our architecture using three data sets extracted from the DCASE 2016-2018 challenges [11]. The data sets contain 1,170, 4,680, and 8,640 audio scenes with lengths between 10 and 30 seconds. We use only the DCASE development data and label the data sets according to their source, *scenes16*, *scenes17*, and *scenes18*. There are 15 different classes in *scenes16* and *scenes17*, 10 in *scenes18*. See the labels in Figure 2 for a complete list of all scenes.

To train our encoder, we randomly select 80% of the scenes in each data set for training and the rest for testing. To generate audio

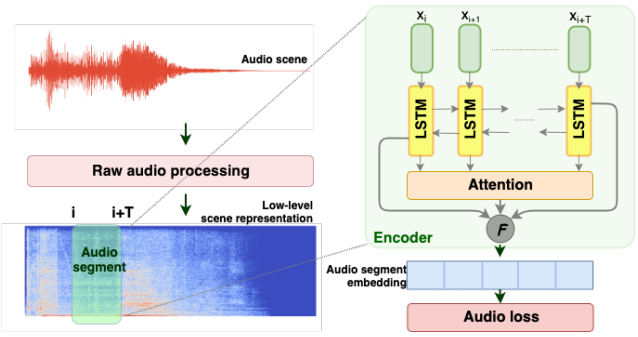

Figure 1: Our audio scene classification consists of three main components: Raw audio processing uses signal processing transforms to extract basic audio features (Section 3.1); the encoder computes embeddings for short duration audio segments (Section 3.2); we train the encoder by optimizing a custom audio loss function (Section 3.3).

segments, we first run FFT on windows of 25ms with 10ms overlap in each scene and compute the low-level features for each window (Section 3.1). An audio segment consists of 20 consecutive FFT windows (or around 300ms of real audio). We select triplets using semi-hard negative sampling, *i.e.*, anchors are selected at random and a negative sample is never closer to the anchor than the positive sample [22]. We run the Adam optimizer with a learning rate of 0.001, batch size of 64, and audio loss regularizer of 0.1. All results are computed on the test data and capture averages over at least five experiments.

We measure the detection accuracy in two ways. To provide a baseline, we compute the *all segment* prediction: we classify a scene with majority voting across *all* its segments. Our contribution is the *first k segments* prediction: we classify a scene through majority voting only on its *first k* segments. To understand the limits of early detection, in practice we set k to 1 to obtain the *first segment* accuracy. To compute the accuracy of a single segment we use nearest-neighbor based classification: we classify a test segment in the same class as its nearest neighbor in embedding space.

4.2 Results

Table 1 summarizes the accuracy results for all data sets. Focus on the first two lines for now. The first segment prediction (*i.e.*, k=1) conserves much of the accuracy obtained when hearing the entire scene: listening to the first 300ms (*i.e.*, the duration of an audio segment) instead of the entire scene (of at least 10s) reduces the detection accuracy by only 7%.

Because our goal is efficient audio detection based on processing the first few segments rather than the entire scene, a direct comparison to other work classifying entire scenes [7, 14, 15] is not pertinent. Nevertheless, we note that we achieve similar results on entire scenes in the DCASE data as Guo *et al.* [7] (75.01% vs 74.90% for their non-ensemble methods). They compute a single embedding for an entire six second audio scene, which means they must hear the whole scene before classifying it.

We also compute the accuracy when increasing the number of segments at the beginning of a scene that we listen to. Figure 3(top)

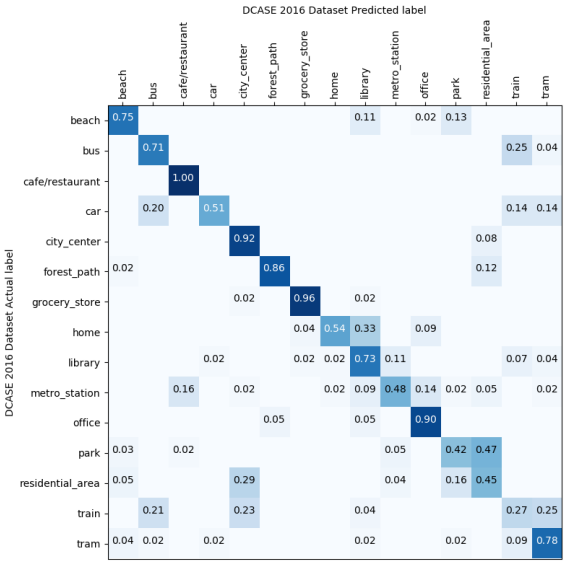

Figure 2: The confusion matrix for the first segment approach on the *scenes16* data set.

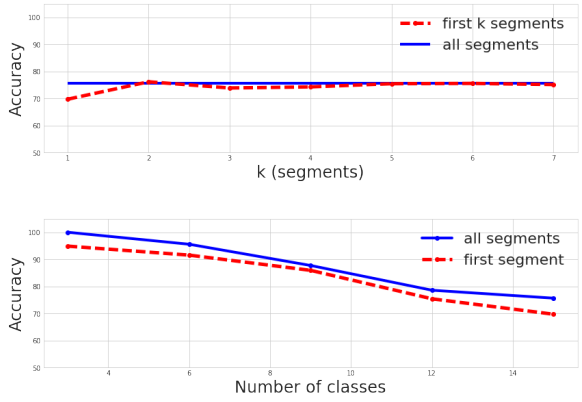

Figure 3: Average accuracy the *scenes16* data set as we vary the number of segments we listen to at the beginning of each scene (top) and the number of possible classes of each scene (bottom). The duration of a segment is approximately 300ms.

presents the results for *scenes16*. We are able to match the *all segments* accuracy after hearing only the first two segments (or 1s) of a scene. Although the *first k segments* accuracy drops slightly with $k > 2$, its value is still close to the *all segment* accuracy. We conclude that we can reliably detect the category of a scene after less than one second.

Figure 3(bottom) plots the detection accuracy as we vary the total number of scene classes. To obtain these results, we randomly select scene classes and train only on samples from these classes. Our results show that the fewer the scenes to choose from, the better our framework can predict the correct scene: we obtain 95% accuracy when looking at the first segment only and with two scenes to choose from.

	scenes16	scenes17	scenes18
First segment	70.28	67.25	61.92
All segments	75.01	69.98	67.15
First segment (triplet)	68.83	65.26	61.27
All segments (triplet)	73.23	68.11	65.87
First segment (no attn)	65.51	63.62	61.38
All segments (no attn)	69.96	66.96	64.01

Table 1: Detection accuracy results. We compare the *all segments* and *first segment* predictions with first segment predictions when using a simple loss function, as in Equation 6 ("triplet") and without an attention layer ("no attn").

Are some scenes more accurately predicted than others? We compute the *first segment* accuracy for each individual class and present the results as a confusion matrix in Figure 2. While most classes have a high accuracy, we notice that we cannot always differentiate scenes with similar sounds (*e.g.*, tram and train). This is because embeddings of segments in these scenes are close to each other and our nearest neighbor classifier cannot tell them apart. We are investigating non-linear classifiers to discriminate segments in such scenarios.

Finally, we seek to better understand the role of the audio loss and attention in emphasizing the relevant segments in a scene. We re-run the experiments as follows: first, by replacing the audio loss with a simple triplet loss (Equation 6) ("triplet"); second, after removing the attention layer ("no attn"). In both cases, we also remove the *importance* embedding; the *similarity* embedding becomes the final embedding. Table 1 presents the results. The audio loss can improve classification even when we listen to the entire scene. Furthermore, without audio loss or attention, the first segment classification becomes worse: accuracy drops to 65%, compared to 75% for *all segments* and 70% for *first segment* when both attention and audio loss are present.

The improvements from attention and audio loss are higher in the *scenes16* and *scenes17* than in *scenes18*. Since the *first segment* results are similar in *scenes18*, it is likely that a single segment does not have enough information to discriminate between classes that are acoustically close. When adding more segments, the effect of attention and audio loss is more visible, albeit not as much as for *scenes16* and *scenes17*. A potential reason for this is the much wider diversity of the *scenes18* data, which contains scenes recorded in six cities, instead of the two for the other two data sets.

5 CONCLUSIONS

We introduced a deep learning based audio classification framework that can detect the category of the environment with high accuracy after listening in for at most 300ms. Our framework relies on jointly learning embeddings for short audio segments, including an *importance* embedding associated with each segment. These embeddings help identify the relevant segments within a scene and underplay the nondescript segments, and underline the important time steps within a segment. Two ideas are critical to our approach: formulating the classification problem as a short audio segment retrieval problem and constructing a new learning optimization objective that captures segment similarity and importance.

REFERENCES

[1] Sumair Aziz, Muhammad Awais, Talha Akram, Muhammad Umar, Khursheed Khursheed, and Musaed Alhussein. 2019. Automatic Scene Recognition through Acoustic Classification for Behavioral Robotics. *Electronics* 8 (04 2019). https://doi.org/10.3390/electronics8050483

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR* abs/1409.0473 (2014).

[3] Pedro Cano, Eloi Batlle, Ton Kalker, and Jaap Haitsma. 2005. A Review of Audio Fingerprinting. *J. VLSI Signal Process. Syst.* 41, 3 (Nov. 2005), 271–284.

[4] Gianmarco Cerutti, Rahul Prasad, Alessio Brutti, and Elisabetta Farella. 2019. Neural Network Distillation on IoT Platforms for Sound Event Detection. (08 2019).

[5] Shizhe Chen, Jia Chen, Qin Jin, and Alexander Hauptmann. 2018. Class-aware Self-Attention for Audio Event Recognition. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval* (Yokohama, Japan) *(ICMR '18)*. ACM, New York, NY, USA, 28–36. https://doi.org/10.1145/3206025.3206067

[6] Xuerui Dai and Xueye Wei. 2018. HybridNet: A fast vehicle detection system for autonomous driving. *Signal Processing: Image Communication* 70 (09 2018). https://doi.org/10.1016/j.image.2018.09.002

[7] Jinxi Guo, Ning Xu, Li-Jia Li, and Abeer Alwan. 2017. Attention Based CLDNNs for Short-Duration Acoustic Scene Classification. In *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017*. 469–473. http://www.isca-speech.org/archive/Interspeech_2017/abstracts/0440.html

[8] D. Istrate, E. Castelli, M. Vacher, L. Besacier, and J. F. Serignat. 2006. Information Extraction from Sound for Medical Telemonitoring. *Trans. Info. Tech. Biomed.* 10, 2 (April 2006).

[9] Aren Jansen, Manoj Plakal, Ratheet Pandya, Daniel P. W. Ellis, Shawn Hershey, Jiayang Liu, R. Channing Moore, and Rif A. Saurous. 2017. Unsupervised Learning of Semantic Audio Representations. *CoRR* abs/1711.02209 (2017). arXiv:1711.02209 http://arxiv.org/abs/1711.02209

[10] Alfred Mertins and Dr Alfred Mertins. 1999. *Signal Analysis: Wavelets, Filter Banks, Time-Frequency Transforms and Applications*. John Wiley & Sons, Inc., New York, NY, USA.

[11] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. 2016. TUT Database for Acoustic Scene Classification and Sound Event Detection. In *24th European Signal Processing Conference 2016 (EUSIPCO 2016)*. Budapest, Hungary.

[12] Arthur William Moore and James W. Jorgenson. 1993. Median filtering for removal of low-frequency background drift. *Analytical chemistry* 65 2 (1993), 188–91.

[13] Nobutaka Ono, Kenichi Miyamoto, Jonathan Le Roux, Hirokazu Kameoka, and Shigeki Sagayama. 2008. Separation of a monaural audio signal into harmonic/percussive components by complementary diffusion on spectrogram. (01 2008).

[14] Huy Phan, Oliver Y. Chén, Lam Dang Pham, Philipp Koch, Maarten De Vos, Ian Vince McLoughlin, and Alfred Mertins. 2019. Spatio-Temporal Attention Pooling for Audio Scene Classification. *CoRR* abs/1904.03543 (2019). arXiv:1904.03543 http://arxiv.org/abs/1904.03543

[15] Huy Phan, Philipp Koch, Fabrice Katzberg, Marco Maaß, Radoslaw Mazur, and Alfred Mertins. 2017. Audio Scene Classification with Deep Recurrent Neural Networks. In *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017*. 3043–3047. http://www.isca-speech.org/archive/Interspeech_2017/abstracts/0101.html

[16] Aref Pour, Mohammad Asgari, and Mohammad Hasanabadi. 2014. Gammatonegram based speaker identification. *Proceedings of the 4th International Conference on Computer and Knowledge Engineering, ICCKE 2014*, 52–55.

[17] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 815–823.

[18] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'14)*.

[19] Nicolas Turpault, Romain Serizel, and Emmanuel Vincent. 2019. Semi-supervised Triplet Loss Based Learning of Ambient Audio Embeddings. 760–764. https://doi.org/10.1109/ICASSP.2019.8683774

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., USA, 6000–6010. http://dl.acm.org/citation.cfm?id=3295222.3295349

[21] Wired 2017. Wired. https://www.wired.com/story/driverless-cars-need-ears-as-well-as-eyes/.

[22] Chao-Yuan Wu, R. Manmatha, Alexander J. Smola, and Philipp Krähenbühl. 2017. Sampling Matters in Deep Embedding Learning. *CoRR* abs/1706.07567 (2017). arXiv:1706.07567 http://arxiv.org/abs/1706.07567

[23] Hong-Bo Zhang, Yi-Xiang Zhang, Bineng Zhong, Qing Lei, Lijie Yang, Ji-Xiang Du, and Duan-Sheng Chen. 2019. A Comprehensive Survey of Vision-Based Human Action Recognition Methods. In *Sensors*.

[24] Fang Zheng, Guoliang Zhang, and Zhanjiang Song. 2001. Comparison of different implementations of MFCC. *Journal of Computer Science and Technology* 16, 6 (01 Nov 2001), 582–589. https://doi.org/10.1007/BF02943243