BEYOND EQUAL-LENGTH SNIPPETS: HOW LONG IS SUFFICIENT TO RECOGNIZE AN AUDIO SCENE?

Huy Phan[*]*, Oliver Y. Chén*[*]*, Philipp Koch*[†]*, Lam Pham*[‡]
Ian McLoughlin[‡]*, Alfred Mertins*[†]*, and Maarten De Vos*[*]

[*] University of Oxford, Department of Engineering Science, UK
[‡] University of Kent, School of Computing, UK
[†] University of Lübeck, Institute for Signal Processing, Germany

ABSTRACT

Due to the variability in characteristics of audio scenes, some can naturally be recognized earlier, i.e. after a shorter duration, than others. In this work, rather than using equal-length snippets for all scene categories, as is common in the literature, we study to which temporal extent an audio scene can be reliably recognized. For modelling, in addition to two single-network systems relying on a convolutional neural network and a recurrent neural network, we also investigate early fusion and late fusion of these two single networks for audio scene classification. Moreover, as model fusion is prevalent in audio scene classifiers, we further aim to study whether and when model fusion is really necessary for this task.

Index Terms— Audio scene classification, convolutional neural network, recurrent neural network, early fusion, late fusion

1. INTRODUCTION

Audio scene recognition (ASC) [1, 2] is an important task in machine hearing [3, 4]. A common goal is to replicate the abilities of human hearing in recognizing environments based on acoustic signals. Over the past few years, research on this problem has advanced rapidly in terms of performance improvement [5, 6, 7, 8, 9] as well as the number of available datasets [10, 11, 12, 13].

Audio scenes vary significantly in their characteristics, i.e. background noise and foreground sound events, and therefore the duration required for human ears to perceive and recognize them is different from scene to scene. For instance, one might well recognize an "in airplane" acoustic environment within a few seconds due to its loud and distinguishable background noise. However, it may take minutes of listening to accumulate sufficient acoustic cues to differentiate a "restaurant" from a "cafe" or even a "busy street" with similar babble noise. Plausibly, this variability should also be generalizable to a machine hearing system. However, most, if not all, available ASC datasets assume a fixed signal length for every scene category and instance. Typical lengths of 30 [11, 13] or 10 seconds [10, 12] have been commonly selected when designing such datasets. There also exist studies with shorter signal duration, such as six seconds in [14]. Although a fixed common length facilitates dataset design and experimentation, it does not reflect the reality of the ASC task. Hence this paper investigates how long is sufficient to reliably recognize different acoustic scenes given state-of-the-art models. These findings would also suggest how an ASC could be implemented in real applications in practice. That is, for such a system, the listening duration should be adapted to different kinds of scenes in order to reach a certain level of recognition certainty.

Methodology-wise, approaches based on convolutional neural networks (CNNs) [6, 15, 2, 16, 7] and recurrent neural networks (RNNs) [5, 17, 9] have been demonstrated to be most efficient for ASC. In addition, state-of-the-art performance has often been achieved with ensembles of multiple networks [8, 18, 19]. Therefore, in this study, we tackle ASC with two single-network models based on CNN and RNN as well as investigate two fusion schemes of these models. An early fusion scheme constructs a two-stream convolutional-recurrent neural network (C-RNN) and explores in-network fusion of the two streams' learned features before classification. A late fusion scheme trains two standalone networks independently and probabilistically aggregates their classification results at the end. Although ensemble methods are well-established in improving performance of machine learning systems, and have been applied to ASC as a rule of thumb, no prior work has studied whether and to which extent model fusion is necessary and most useful for the ASC task. Hence we also seek to answer that question.

2. CNN AND RNN FOR ASC

The CNN model proposed here improves on the one in [2] in that its convolutional filters are trained to take into account invariance across multiple input feature channels. The RNN model is based on [5], which reported state-of-the-art performance on the LITIS-Rouen dataset [13]. Both the CNN and RNN models share common input features.

2.1. Input features

An audio snippet is first decomposed into small frames of 250 ms with 50% overlap. We employed the label tree embedding (LTE) features proposed in [20] to represent an audio frame. To this end, low-level feature vectors are first extracted for the audio frames and used to construct a label tree. Given C scene categories, the constructed label tree consists of $C-1$ split nodes which index $2(C-1)$ meta-classes in their left and right child nodes. A low-level feature vector of an audio frame is then mapped into an LTE feature vector $\mathbf{x} \in [0,1]^F$, $F = 2(C-1)$, whose entries encode the posterior probabilities of the audio frame belonging to the meta-classes. As a result, we obtain T such LTE feature vectors for the audio snippet consisting of T frames.

In that way, we employ three different low-level feature sets: (1) Gammatone spectral coefficients [21], (2) MFCCs, and (3) log-

The research was supported by the NIHR Oxford Biomedical Research Centre. Corresponding author: huy.phan@eng.ox.ac.uk

frequency filter bank coefficients. We extract LTE features with the presence/absence of the background noise as in [5, 2]. In total, we obtain $D = 6$ feature channels so the audio snippet is eventually represented by a multi-channel image feature $\mathbf{X} \in [0,1]^{F \times T \times D}$.

2.2. CNN

Audio scene recognition using CNNs on LTE features have been previously explored in [2]. However, in that work, over-time convolution with 3-dimensional convolutional filters, that fully cover feature and channel dimensions [2], did not explore the invariance across LTE channels. As an improvement, the CNN proposed here is designed to have 2-dimensional convolutional filters to perform convolution over time as well as across input feature channels.

Let us denote such a 2-dimensional filter as $\mathbf{w} \in \mathbb{R}^{F \times w}$ where $w < T$ represents the temporal width of the filter. Convolving the filter $\mathbf{w}$ with the multi-channel image input $\mathbf{X}$ over-time and across-channel results in a 2-dimensional feature map $\mathbf{O} \in \mathbb{R}^{(T-w+1) \times D}$ whose entries are given by

$$o_{ij} = (\mathbf{X} * \mathbf{w})_{ij} = \sum_{m,n} (\mathbf{X}_j[i : i+w-1] \odot \mathbf{w})_{m,n}. \quad (1)$$

Here, $*$ and $\odot$ indicate the convolution and element-wise multiplication operations, respectively. $\mathbf{X}_j[i : i+w-1]$ denotes an image slice from time index i to $i+w-1$ on the channel index j. *Rectified Linear Units* (ReLU) activation [22] is then applied, followed by 1-max pooling [23, 24] on the 2D feature map to retain the most prominent feature:

$$z^{\text{conv}} = \max_{i,j} o_{ij}. \quad (2)$$

Similar to [2], we design the CNN to have $R = 3$ filter sets corresponding to three temporal widths $w \in \{3, 5, 7\}$ with each filter set consisting of $Q = 1000$ convolutional filters with the same width. The total number of filters is, therefore, $R \times Q$ which leads to a convolutional feature vector $\mathbf{z}^{\text{conv}} \in \mathbb{R}^{RQ}$ after the pooling layer.

During network training, the convolutional feature vector $\mathbf{z}^{\text{conv}}$ is presented to a softmax layer for classification. The CNN is trained to minimize the cross-entropy error over the training examples:

$$E(\boldsymbol{\theta}) = -\sum_{n} \mathbf{y}_n \log\left(\hat{\mathbf{y}}_n(\mathbf{X}_n, \boldsymbol{\theta})\right) + \frac{\lambda}{2} \|\boldsymbol{\theta}\|_2^2. \quad (3)$$

Here, $\boldsymbol{\theta}$ denotes the network parameters and the λ denotes the hyper-parameter of the ℓ_2-norm regularization term. For further regularization, dropout [25] is also applied to the CNN feature map.

After network training, we extract the convolutional feature vector $\mathbf{z}^{\text{conv}}$ and train a linear SVM for classification, in replacement of the softmax as in [2, 5].

2.3. RNN

For the RNN, we stack different channels of the multi-channel image input $\mathbf{X}$ in the feature dimension and treat it as a temporal sequence of feature vectors $(\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots, \tilde{\mathbf{x}}_T)$ where each $\tilde{\mathbf{x}}_i \in \mathbb{R}^{FD}$, $1 \leq t \leq T$. The RNN then reads the input sequence into the sequence of recurrent output vectors $(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_T)$, where

$$\mathbf{z}_t = \mathbf{h}_t \mathbf{W}_z + \mathbf{b}_z, \quad (4)$$

$$\mathbf{h}_t = \mathcal{H}(\tilde{\mathbf{x}}_t, \mathbf{h}_{t-1}). \quad (5)$$

$\mathbf{h}_t \in \mathbb{R}^H$ denotes the hidden state vector of size H at time step t, $\mathbf{W}_z \in \mathbb{R}^{H \times H}$ is a weight matrix and $\mathbf{b}_z \in \mathbb{R}^H$ denotes a bias term. $\mathcal{H}$ represents the hidden layer function of the recurrent layer and is realized using a Gated Recurrent Unit (GRU) [26]. The RNN is designed to have two recurrent layers with the hidden state vector size of $H = 256$. The recurrent layers are stacked on top of each other to construct a deep RNN similar to [5, 27].

We eventually retain the output vector at the last time index T as the recurrent feature vector, i.e. $\mathbf{z}^{\text{rec}} \equiv \mathbf{z}_T \in \mathbb{R}^H$, as it is expected to have encoded information of the entire input sequence. Dropout is also applied to the recurrent feature vector for regularization. Similarly to the CNN case, softmax and the cross-entropy loss given in (3) are employed for classification during network training. After training, the recurrent features $\mathbf{z}^{\text{rec}}$ are also extracted and used to train a linear SVM classifier for classification.

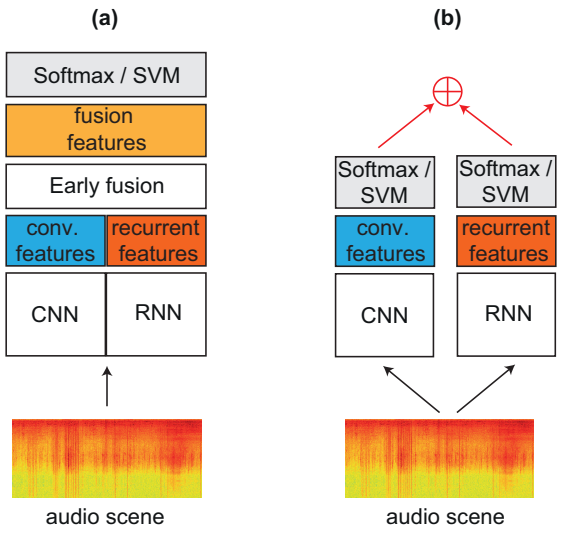

Fig. 1. (a) Early fusion with two-stream C-RNN and (b) late fusion with two networks, CNN and RNN.

3. EARLY FUSION WITH TWO-STREAM C-RNN

For the early fusion scheme, we construct a two-stream C-RNN network as illustrated in Fig. 1(a) and fuse the features learned by the CNN and RNN streams before the classification taken place. The idea is to allow the network to explore combinations of two feature types to optimize the classification task. The CNN and RNN streams in the C-RNN network have the same body architectures as the CNN and RNN described in Sections 2.2 and 2.3.

At the fusion layer of the C-RNN network, a fusion function $f : (\mathbf{z}^{\text{conv}}, \mathbf{z}^{\text{rec}}) \mapsto \mathbf{z}^f$ fuses the convolutional feature vector $\mathbf{z}^{\text{conv}}$ of the CNN stream and the recurrent feature vector $\mathbf{z}^{\text{rec}}$ of the RNN stream to produce an output map $\mathbf{z}^f$. We investigate the following fusion functions f: *sum fusion*, *max fusion*, and *concatenation fusion*. In addition, as $\mathbf{z}^{\text{conv}}$ and $\mathbf{z}^{\text{rec}}$ differ in size, we transform them via a fully-connected layer with *sigmoid* activation beforehand to make their sizes compatible (i.e. with sum fusion and max fusion) or to equalize their contributions to the output map (i.e. with concatenation fusion);

$$\tilde{\mathbf{z}}^{\text{conv}} = \mathbf{z}^{\text{conv}} \mathbf{W}^{\text{conv}} + \mathbf{b}^{\text{conv}}, \quad (6)$$

$$\tilde{\mathbf{z}}^{\text{rec}} = \mathbf{z}^{\text{rec}} \mathbf{W}^{\text{rec}} + \mathbf{b}^{\text{rec}}. \quad (7)$$

Here, $\mathbf{W}^{\text{conv}} \in \mathbb{R}^{RQ \times M}$ and $\mathbf{W}^{\text{rec}} \in \mathbb{R}^{H \times M}$ denote weight matrices while $\mathbf{b}^{\text{conv}} \in \mathbb{R}^M$ and $\mathbf{b}^{\text{rec}} \in \mathbb{R}^M$ denotes bias terms of the

fully-connected layers. M is the desired size of the transformed feature vectors $\tilde{\mathbf{z}}^{\text{conv}}$ and $\tilde{\mathbf{z}}^{\text{rec}}$.

Sum fusion. $\mathbf{z}^{f_{\text{sum}}} = f_{\text{sum}}(\tilde{\mathbf{z}}^{\text{conv}}, \tilde{\mathbf{z}}^{\text{rec}})$ computes the sum of the two feature vectors $\mathbf{z}^{f_{\text{sum}}} = \tilde{\mathbf{z}}^{\text{conv}} + \tilde{\mathbf{z}}^{\text{rec}}$.

Max fusion. $\mathbf{z}^{f_{\text{max}}} = f_{\text{max}}(\tilde{\mathbf{z}}^{\text{conv}}, \tilde{\mathbf{z}}^{\text{rec}})$ takes the maximum of the two feature vectors $z_i^{f_{\text{max}}} = \max(\tilde{z}_i^{\text{conv}}, \tilde{z}_i^{\text{rec}}), 1 \leq i \leq M$.

Concatenation fusion. $\mathbf{z}^{f_{\text{cat}}} = f_{\text{cat}}(\tilde{\mathbf{z}}^{\text{conv}}, \tilde{\mathbf{z}}^{\text{rec}})$ simply concatenates the two feature vectors to make a larger one $\mathbf{z}^{f_{\text{cat}}} = [\tilde{\mathbf{z}}^{\text{conv}}, \tilde{\mathbf{z}}^{\text{rec}}]$.

Network training and evaluation were performed similarly to the standalone CNN and RNN with a softmax layer used for classification and minimization of the cross-entropy loss given in (3). In particular, dropout is further applied to the fusion features for regularization purpose. Again, the fusion features are extracted to train a linear SVM for classification after network training.

4. LATE FUSION OF CNN AND RNN

For the late fusion scheme, we probabilistically combine the classification results of the standalone CNN and RNN in Sections 2.2 and 2.3, respectively, as illustrated in Fig. 1(b). We study three fusion methods: *max fusion*, *mean fusion*, and *multiplication fusion*. Let $\mathbf{P}^{\text{conv}} = (P_1^{\text{conv}}, P_2^{\text{conv}}, \ldots, P_C^{\text{conv}})$ and $\mathbf{P}^{\text{rec}} = (P_1^{\text{rec}}, P_2^{\text{rec}}, \ldots, P_C^{\text{rec}})$ denote the posterior probabilities obtained by the CNN and RNN, respectively. The classification likelihood $\mathbf{P} = (P_1, P_2, \ldots, P_C)$ after fusion with the three methods is then given by

$$P_c = \max(P_c^{\text{conv}}, P_c^{\text{rec}}) \qquad \text{for } 1 \leq c \leq C, \tag{8}$$

$$P_c = \frac{1}{2}(P_c^{\text{conv}} + P_c^{\text{rec}}) \qquad \text{for } 1 \leq c \leq C, \tag{9}$$

$$P_c = \frac{1}{2}(P_c^{\text{conv}} \times P_c^{\text{rec}}) \qquad \text{for } 1 \leq c \leq C, \tag{10}$$

respectively. The final output label is then determined by likelihood maximization on the classification likelihood $\mathbf{P}$.

5. EXPERIMENTS

5.1. LITIS-Rouen dataset and modification for this study

We conducted experiments using the LITIS-Rouen dataset [13]. This dataset consists of 19 scene categories with 3026 examples in total. All instances have the same length of 30 seconds and were recorded with a sampling rate of 22050 Hz.

However we did not use the full 30-second snippets *per se*. Instead, we decomposed each 30-second snippet into segments of 4 seconds length without overlap (except the last segment). Thus $S = 8$ such segments were obtained from each 30-second snippet. The classification models were trained with 4-second segments extracted from the training data. To understand how the classification performance varies with different test signal lengths, we sequentially evaluated a classification model on the 4-second segments of a 30-second snippet and aggregated the classification results over $\{1, 2, \ldots, S\}$ segments which are equivalent to $\{4, 8, \ldots, 28, 30\}$ seconds. In order to aggregate classification results over multiple segments in the late-fusion system, we used the same fusion method that we used for model fusion. For the standalone CNN, RNN, and the two-stream C-RNN, we employed the multiplication fusion as it has been shown to be efficient for this purpose [28, 5].

For an experiment with a specific test signal length, we follow the training/testing splits in the seminal work [13] with the reported performances averaged over 20 splits.

Table 1. Overall performance of the classification models averaged over all test signal lengths $\{4, 8, \ldots, 28, 30\}$. Early fusion and late fusion are abbreviated by EF and LF, respectively.

Method	Accuracy	F1-score	Precision
CNN	95.7	95.5	95.3
RNN	95.8	95.7	95.5
EF - Sum	94.8	94.5	94.3
EF - Max	94.6	94.3	94.1
EF - Concat	94.8	94.5	94.3
LF - Max	95.5	95.3	95.1
LF - Mean	95.9	95.7	95.5
LF - Mult	96.1	95.9	95.7

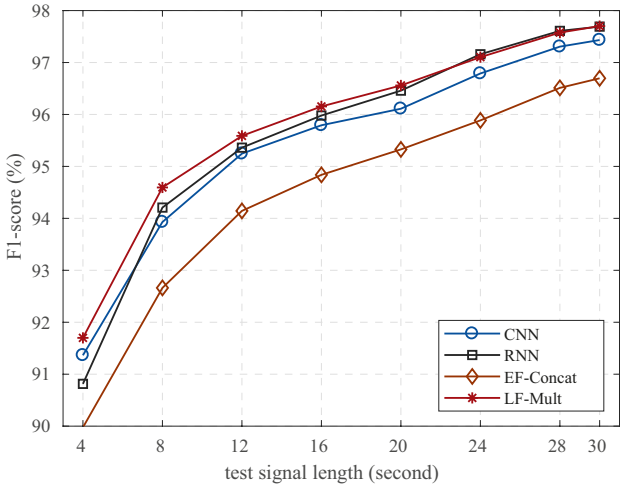

Fig. 2. The variation in overall F1-score over different test signal length for CNN, RNN, EF-Concat and LF-Mult schemes.

5.2. Network training and parameters

A dropout rate of 0.5 and 0.1 was used for the CNN and the RNN, respectively. Particularly, for the early fusion C-RNN network, a dropout rate of 0.5 was further applied to the fusion features. For all networks, the regularization parameter λ was commonly set to 10^{-3} and the network training was accomplished using the *Adam* optimizer [29] with a learning rate of 10^{-4}. In addition, the hyper-parameter C of the SVMs used for classification after network training was fixed to 1.

5.3. Experimental results

5.3.1. Early fusion or late fusion

Table 1 shows the overall performance obtained by the studied classification systems averaged over different test signal lengths $\{4, 8, \ldots, 28, 30\}$. Fig. 2 further shed light on the variation of the their F1-score over the spectrum of the test signal length. For clarity, we only show the best early-fusion system (i.e. EF - Concat) and the best late-fusion system (i.e. LF - Mult) in the figure.

On average, the standalone CNN marginally underperforms its RNN counterpart. Their performance gap is particularly noticeable with large test signal lengths ($\geq$ 20 seconds). These results highlight the importance of sequential modelling for temporal data. It

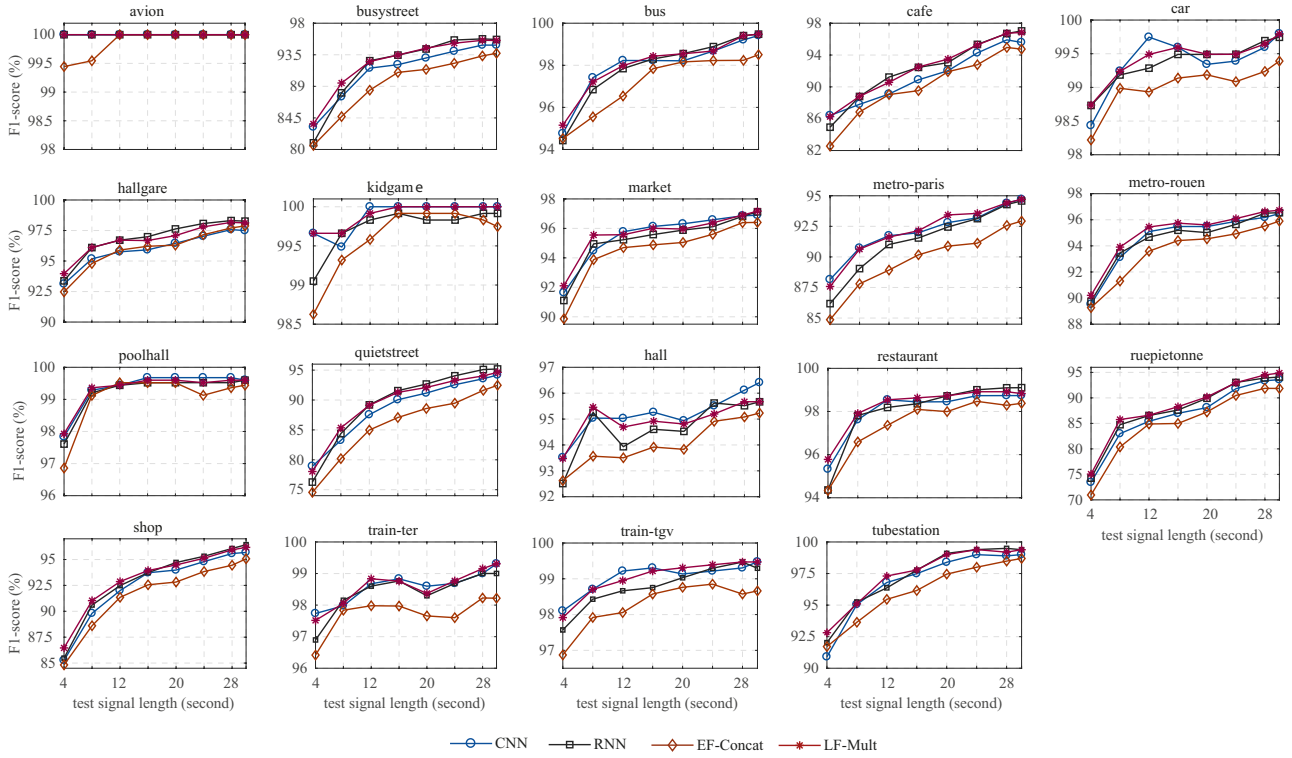

Fig. 3. The variation in category-specific F1-score over the range of test signal lengths for the same systems as in Fig. 2.

should also be noticed that the CNN proposed here achieves an F1-score of 97.4%, improving that of the over-time-convolution CNN in [2] (i.e. 96.5%) by 0.9% absolute. Regarding the fusion systems, sum fusion and concatenation fusion perform comparably in early fusion and appear to be better than max fusion. However, among the late fusion methods, multiplication fusion performs best. Compared to other methods, multiplication fusion favours likelihood for categories that have consistent classification results, and more aggressively suppresses those with diverged classification results [28, 5].

More importantly, the results in Table 1 and Fig. 2 show that early fusion consistently worsens the classification performance. On average, the best early-fusion system (i.e. EF - Concat) reduces the F1-score by 1.0% and 1.2% absolute compared to the standalone CNN and RNN, respectively. On the contrary, late fusion is more efficient as LF - Mult yields gains of 0.4% and 0.2% absolute on average F1-score compared to the standalone CNN and RNN, which was previously reported with state-of-the-art performance at 30-second test signal length on the experimental dataset [5].

5.3.2. *When is model fusion useful?*

Inspection of Fig. 2 further reveals that model fusion is most productive when the test signal length is small (i.e. < 20 seconds). When the test signal length is larger than 20 seconds, the performance gain (if any) achieved by LF - Mult is very marginal. Intuitively, with short signal lengths, multiple views on a short duration scene can compensate each other in the ensemble. However, when listening longer, the best standalone system (namely RNN) has accumulated more information about a scene and can recognize it quite reliably. Meanwhile the weaker standalone model (namely the CNN), does not appear to bring sufficient new information about the scene into the ensemble.

5.3.3. *How long is sufficient to recognize a scene?*

Fig. 3 shows the F1-score variation patterns for several different scene categories over a range of test signal lengths. It is unsurprising to find that the patterns are very category-specific. Several categories can be reliably recognized even with a very short signal length. For example, we achieve perfect or nearly perfect recognition accuracy rate on "avion", "kidgame", and "poolhall" within 4, 12, and 16 seconds, respectively. Some other scenes, such as "car", "metro-rouen", "restaurant", "train-ter", and "train-tgv", can also be recognized reliably within 16 seconds, and accumulating more information about these scenes does not enable significant gains in performance. By contrast, some other scenes require much longer test signal lengths, e.g. 30 seconds, to achieve good performance. Examples include "cafe", "quitestreet", "ruepietonne", and "shop". It would be expected that a system should listen even longer (i.e. > 30 seconds) to improve the detection certainty for those scenes.

6. CONCLUSION

Rather than assuming a fixed duration for every scene category as in much of the literature, this work studied an important aspect of the task, namely how much time is sufficient to recognize an audio scene. This was accomplished by exploring ASC using a CNN, an RNN and their fusion using various early and late fusion strategies. We additionally investigated which kind of model fusion is most useful for the task. Experimental results on the LITIS-Rouen dataset showed that the duration required to recognize a scene is category-specific, just as it is for human listeners. Some scenes were reliably recognized within a few seconds while significantly longer durations were necessary to recognize others. Furthermore, while late fusion was shown to improve the classification performance, it is most beneficial when the test signal length is limited.

7. REFERENCES

[1] Dan Stowell, Dimitrios Giannoulis, Emmanouil Benetos, Mathieu Lagrange, and Mark D. Plumbley, "Detection and classification of acoustic scenes and events," *IEEE Trans. Multimedia*, vol. 17, no. 10, pp. 1733–1746, 2015.

[2] H. Phan, L. Hertel, M. Maass, P. Koch, R. Mazur, and A. Mertins, "Improved audio scene classification based on label-tree embeddings and convolutional neural networks," *IEEE/ACM Trans. on Audio, Speech and Language Processing (TASLP)*, vol. 25, no. 6, pp. 1278–1290, 2017.

[3] R. F. Lyon, "Machine hearing: An emerging field," *IEEE Signal Processing Magazine*, vol. 27, no. 5, pp. 131–139, 2010.

[4] D. Wang and G. J. Brown, *Computational Auditory Scene Analysis: Principles, Algorithms, and Applications*, Wiley-IEEE Press, 2006.

[5] H. Phan, P. Koch, F. Katzberg, M. Maass, R. Mazur, and A. Mertins, "Audio scene classification with deep recurrent neural networks," in *Proc. Interspeech*, 2017, pp. 3043–3047.

[6] Hamid Eghbal-Zadeh, Bernhard Lehner, Matthias Dorfer, and Gerhard Widmer, "CP-JKU submissions for DCASE-2016: a hybrid approach using binaural I-vectors and deep convolutional neural networks," Tech. Rep., Detection and Classification of Acoustic Scenes and Events 2016, 2016.

[7] S. Mun, S. Park1, D. Han, and H. Ko, "Generative adversarial network based acoustic scene training set augmentation and selection using SVM hyper-plane," in *Proc. Workshop on Detection and Classification of Acoustic Scenes and Events*, 2017.

[8] Y. Sakashita and M. Aono, "Acoustic scene classification by ensemble of spectrograms based on adaptive temporal divisions," Tech. Rep., DCASE2018 Challenge, 2018.

[9] T. Zhang, K. Zhang, and J. Wu, "Temporal transformer networks for acoustic scene classification," in *Proc. Interspeech 2018*, 2018, pp. 1349–1353.

[10] A. Mesaros, T. Heittola, and T. Virtanen, "A multi-device dataset for urban acoustic scene classification," *arXiv preprint arXiv:1807.09840*, 2018.

[11] A. Mesaros, T. Heittola, E. Benetos, P. Foster, M. Lagrange, T. Virtanen, and M. D. Plumbley, "Detection and classification of acoustic scenes and events: Outcome of the DCASE 2016 challenge," *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 26, no. 2, pp. 379–393, 2018.

[12] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "DCASE 2017 challenge setup: Tasks, datasets and baseline system," in *Proc. Workshop on Detection and Classification of Acoustic Scenes and Events*, 2017.

[13] A. Rakotomamonjy and G. Gasso, "Histogram of gradients of time-frequency representations for audio scene classification," *IEEE/ACM Trans. Audio, Speech, and Language Processing*, vol. 23, no. 1, pp. 142–153, 2015.

[14] J. Guo, N. Xu, L.-J. Li, and A. Alwan, "Attention based CLDNNs for short-duration acoustic scene classification," in *Proc. Interspeech*, 2017.

[15] M. Valenti, A. Diment, G. Parascandolo, S. Squartini, and T. Virtanen, "DCASE 2016 acoustic scene classification using convolutional neural networks," Tech. Rep., Detection and Classification of Acoustic Scenes and Events 2016, 2016.

[16] H. Phan, P. Koch, L. Hertel, M. Maass, R. Mazur, and A. Mertins, "CNN-LTE: a class of 1-X pooling convolutional neural networks on label tree embeddings for audio scene classification," in *Proc. ICASSP*, 2017.

[17] T. H. Vu and J.-C. Wang, "Acoustic scene and event recognition using recurrent neural networks," Tech. Rep., Detection and Classification of Acoustic Scenes and Events 2016, 2016.

[18] S. H. Bae, I. Choi, and N. S. Kim, "Acoustic scene classification using parallel combination of LSTM and CNN," Tech. Rep., Detection and Classification of Acoustic Scenes and Events 2016, 2016.

[19] Y. Yin, R. R. Shah, and R. Zimmermann, "Learning and fusing multimodal deep features for acoustic scene categorization," in *Proc. ACMMM*, 2018, pp. 1892–1900.

[20] H. Phan, L. Hertel, M. Maass, P. Koch, and A. Mertins, "Label tree embeddings for acoustic scene classification," in *Proc. ACMMM*, 2016, pp. 486–490.

[21] D. P. W. Ellis, "Gammatone-like spectrograms," 2009.

[22] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011, pp. 315–323.

[23] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. EMNLP*, 2014, pp. 1746–1751.

[24] H. Phan, L. Hertel, M. Maass, and A. Mertins, "Robust audio event recognition with 1-max pooling convolutional neural networks," in *Proc. Interspeech*, 2016, pp. 3653–3657.

[25] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research (JMLR)*, vol. 15, pp. 1929–1958, 2014.

[26] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. EMNLP*, 2014, pp. 1724–1734.

[27] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. ICASSP*, 2013, pp. 6645–6649.

[28] Yoonchang Han and Kyogu Lee, "Convolutional neural network with multiple-width frequency-delta data augmentation for acoustic scene classification," Tech. Rep., DCASE2016 Challenge, September 2016.

[29] D. P. Kingma and J. L. Ba, "Adam: a method for stochastic optimization," in *Proc. ICLR*, 2015, pp. 1–13.