ATTENTION-BASED CNN WITH GENERALIZED LABEL TREE EMBEDDING FOR AUDIO SCENE CLASSIFICATION

Huy Phan, Philipp Koch, Fabrice Katzberg, Marco Maass, Radoslaw Mazur, and Alfred Mertins

University of Lübeck, Institute for Signal Processing, Lübeck, Germany

{phan,koch,katzberg,maass,mazur,mertins}@isip.uni-luebeck.de

ABSTRACT

This report presents our audio scene classification systems submitted for Task 1 ("acoustic scene classification") of DCASE 2017 challenge [1]. The systems rely on combinations of generalized label tree embedding representation, convolutional neural networks (CNNs), and attention mechanism. Our experimental results on the development data of the challenge show that our proposed system significantly outperform the challenge's baseline, improving the average classification accuracy from 74.8% of the baseline to 83.8%. However, we achieve significantly lower accuracies on the evaluation data, underperforming the DCASE baseline, due to overfitting caused by cross-validation errors in our submission systems.

Index Terms— audio scene classification, CNN, attention, generalized label tree embeddings

1. INTRODUCTION

Label tree embedding (LTE) [2] has been shown to be efficient in transforming and reducing complex audio scenes into semantic representations to expose their useful patterns. As a result, these representations facilitates training deep networks for recognition, such as template matching with CNNs [3, 4] and sequence modeling with RNNs [5], with good performances reported for audio scene classification (ASC) [3, 4, 5].

In this work, we introduce generalized label tree embedding (GLTE) which is an improved and generalized version of LTE. The idea behind GLTE is to identify ambiguous categories and direct them in both directions during label tree construction rather than forcing them to be split too early at a split node as in the case of LTE. As a result, we expect to obtain a better representation, i.e. GLTE, compared to LTE [2]. We investigate coupling CNNs with the GLTE representation for ASC. First, a CNN similar to those used in [6, 3, 4] for template learning and matching will be explored. We further study to integrate the attention mechanism [7, 8, 9] to this CNN to produce an *attentive* CNN.

In general, an audio scene contains different kinds of foreground events mixed with background noise. The foreground events are usually informative for recognizing a scene [10, 11]. It is, therefore, reasonable to somehow weight different parts of a scene differently in a classification model in hope that the weights will be adapted to focus stronger on those informative parts. Ideally, these weights should be automatically learned by the model. This can be accomplished with an attention layer [7, 8, 9]. Attention mechanism has been commonly used with RNNs [7, 8, 9], however, we will show that it can be integrated with the proposed CNN, thank to the CNN's over-time convolution making this possible.

The overall pipeline of the proposed system is illustrated in Figure 1.

2. GENERALIZED LABEL TREE EMBEDDING

2.1. Learning to construct a generalized label tree

Given a training set $\mathcal{S} = \{(\mathbf{x}_n, c_n)\}_{n=1}^{N}$ where $\mathbf{x} \in \mathbb{R}^M$ denotes a low-level feature vector of size M and $c \in \{1, \dots, C\}$ denotes a class label with C is the number of categories. For convenience, let us denote the label set as $\mathcal{L} = \{1, \dots, C\}$. Our goal is to use $\mathcal{S}$ to recursively construct a label tree to encode the hierarchy of the class labels [2, 3].

Similar to the LTE algorithm proposed in [2, 3], the construction algorithm starts at the root node which is associated with the entire set $\mathcal{L}$. Without loss of generality, let us consider a current split node with a label subset $\ell \subset \mathcal{L}$ and the corresponding sample subset $\mathcal{S}^\ell \subset \mathcal{S}$. We then aim at splitting ℓ into two subsets ℓ^L and ℓ^R that satisfy $\ell^L \neq \emptyset$, $\ell^R \neq \emptyset$, $\ell^L \cup \ell^R = \ell$, and $\ell^L \cap \ell^R = \emptyset$. Among all possible partitions, we want to seek for the optimal one such that ℓ^L and ℓ^R can be separated with as few errors as possible using a binary classifier. To accomplish this, two-fold cross validation is performed on $\mathcal{S}^\ell$. For each fold, we decompose $\mathcal{S}^\ell$ into two halves: $\mathcal{S}^\ell_{\text{train}}$ and $\mathcal{S}^\ell_{\text{eval}}$. The former is used to train a multi-class classifier $\mathcal{M}^\ell$ which is then evaluated on $\mathcal{S}^\ell_{\text{eval}}$ to obtain the confusion matrix $\tilde{\mathbf{A}} \in [0,1]^{|\ell|\times|\ell|}$, where $|\cdot|$ represents cardinality of a set. Each element $\tilde{\mathbf{A}}_{ij}$ of $\tilde{\mathbf{A}}$ is computed by

$$\tilde{\mathbf{A}}_{ij} = \frac{1}{|\mathcal{S}^\ell_{\text{eval},i}|} \sum_{\mathbf{x} \in \mathcal{S}^\ell_{\text{eval},i}} P(j|\mathbf{x}, \mathcal{M}^\ell), \tag{1}$$

where $\mathcal{S}^\ell_{\text{eval},i} \subset \mathcal{S}^\ell_{\text{eval}}$ denotes the set of samples with label i. $P(j|\mathbf{x}, \mathcal{M}^\ell)$ denotes the posterior probability that $\mathcal{M}^\ell$ predicts $\mathbf{x}$ as class j. Hence, $\tilde{\mathbf{A}}_{ij}$ with $i \neq j$ indicates how likely a sample of class i is wrongly predicted to belong to class j by the classifier. The overall confusion matrix $\mathbf{A}$ is computed as the average of the confusion matrices obtained by two-fold cross validation. It is further symmetrized as:

$$\bar{\mathbf{A}} = (\mathbf{A} + \mathbf{A}^\mathsf{T})/2. \tag{2}$$

The optimal partition $\{\ell^L, \ell^R\}$ is then derived to minimize:

$$E(\ell) = \sum_{i,j \in \ell^L} \bar{\mathbf{A}}_{ij} + \sum_{m,n \in \ell^R} \bar{\mathbf{A}}_{mn}. \tag{3}$$

By this, the ambiguous categories tend to be grouped into the same subset and, hence, produce two meta-classes $\{\ell^L, \ell^R\}$ that are easy to separate from one another. Spectral clustering [12] is applied on $\bar{\mathbf{A}}$ to solve a relaxed version of the optimization problem in (3).

In the LTE algorithm in [2, 3], ℓ^L and ℓ^R would be immediately directed to the left and right child nodes, respectively. However,

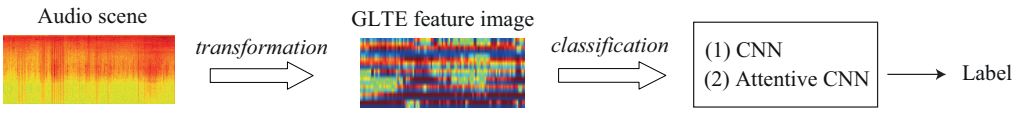

Figure 1: The overall pipeline of the proposed audio scene classification systems.

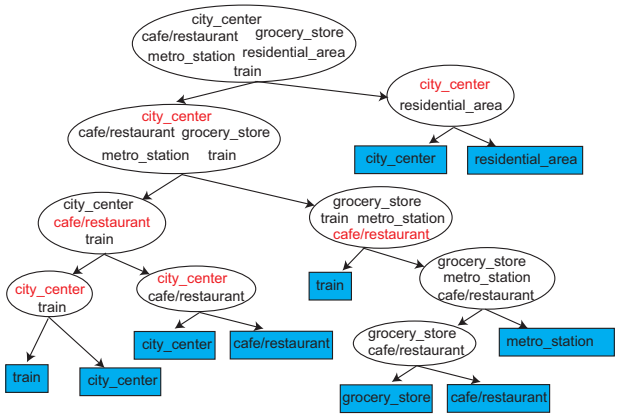

Figure 2: A part of the constructed generalized label tree. The labels in red indicate the ambiguous categories which, therefore, are directed in both left and right child nodes.

this would conservatively enforces categories that are ambiguous to other categories in both ℓ^L and ℓ^R to be split too early and directed in a single direction (i.e. either left or right). They, however, should be sent to both left and right child nodes and be split later with less ambiguity in a deeper level of the tree. That is, they should present in both ℓ^L and ℓ^R. Therefore, we need to identify these ambiguous categories and re-organize the subsets.

We define the *ambiguity* of a category i with categories in a certain label set ℓ as $a(i, \ell)$ which is computed by:

$$a(i, \ell) = \sum_{j \in \ell} \bar{\mathbf{A}}_{ij}. \tag{4}$$

Subsequently, a label $i \in \ell^L$ will be included into ℓ^R if

$$a(i, \ell^R) > \alpha, \tag{5}$$

where $\alpha \in [0, 1]$ denotes the *ambiguity threshold*. A same procedure is applied to all labels in ℓ^R to put them into ℓ^L. Eventually, we obtain two subsets $\{\ell^L, \ell^R\}$ that may have some categories in common. The threshold α is used to regulate the overlapping degree of the subsets and, therefore, the size of the tree. It reduces to the label tree in the LTE case [2, 3] when $\alpha = 1$.

Eventually, ℓ^L and ℓ^R are forwarded to the left and right child nodes, respectively. The splitting process is recursively repeated to grow the whole tree until a leaf node with a single class label is reached. In practice, due to the overlapping, there may exist some nodes with the same label set. During construction, we prune a node which has its label set already existing in the current tree.

We illustrate in Figure 2 a part of the generalized label tree constructed using the DCASE 2017 development data.

2.2. Generalized label tree embedding

GLTE representation can be derived similarly to LTE one in [2, 3]. However, for GLTE, the number of split nodes is dependent on the ambiguity threshold α. Let us denote the number of split nodes as D. We then obtain the embedding $\Psi : \mathbb{R}^M \rightarrow \mathbb{R}^{D \times 2}$ where

$$\Psi(\mathbf{x}) = \big(\psi_1^L(\mathbf{x}), \psi_1^R(\mathbf{x}), \ldots, \psi_D^L(\mathbf{x}), \psi_D^R(\mathbf{x})\big). \tag{6}$$

Here, $\psi_i^L(\mathbf{x})$ and $\psi_i^R(\mathbf{x})$ represent the likelihoods with which the test sample $\mathbf{x}$ belongs to two meta-classes associated with the left and right child nodes of the node i, $1 \leq i \leq D$. To compute $\psi_i^L(\mathbf{x})$ and $\psi_i^R(\mathbf{x})$, at the split node i associated with the label ℓ_i and the partitioning $\{\ell_i^L, \ell_i^R\}$, we train a binary classifier $\mathcal{M}^{\ell_i}$ using the sample set S^{ℓ_i} as training data. We consider the categories with their labels in $\ell_i^L \backslash \ell_i^R$ and $\ell_i^R \backslash \ell_i^L$ as negative class and positive class, respectively. That is, we ignore the common categories in $\ell_i^R \cap \ell_i^L$ when training the classifier $\mathcal{M}^{\ell_i}$. The likelihoods $\psi_i^L(\mathbf{x})$ and $\psi_i^R(\mathbf{x})$ then read

$$\psi_i^L(\mathbf{x}) = P(\text{negative}|\mathbf{x}, \mathcal{M}^{\ell_i}), \tag{7}$$

$$\psi_i^R(\mathbf{x}) = P(\text{positive}|\mathbf{x}, \mathcal{M}^{\ell_i}), \tag{8}$$

where $P(\text{negative}|\mathbf{x}, \mathcal{M}^{\ell_i})$ and $P(\text{positive}|\mathbf{x}, \mathcal{M}^{\ell_i})$ denote the posterior probabilities for classifying the test sample $\mathbf{x}$ into the negative and positive classes, respectively, by the classifier $\mathcal{M}^{\ell_i}$.

Using the embedding, the sample $\mathbf{x}$ is transformed into the high-level representation $\Psi(\mathbf{x})$ (i.e. GLTE presentation).

3. CLASSIFICATION MODELS FOR ASC

3.1. GLTE feature extraction

GLTE representation can be extracted for audio scenes as similarly done for LTE one in [3, 5]. An audio snippet was first decomposed into segments of length 250 ms with 50% overlap. In case of DCASE 2017 challenge, a 10-second snippet yields $T = 78$ segments each of which is represented by a set of low-level features. The per-segment low-level feature vectors were then employed to construct a label tree with the algorithm in Section 2.1. Afterwards, an unseen segment-wise feature vector of the test data was finally transformed into a GLTE feature vector using the embedding in (6).

As in [3, 5], we made use of three low-level feature sets: (1) 64 Gammatone cepstral coefficients extracted in the frequency range of 50 Hz to 22050 Hz, (2) 20 MFCC coefficients, their delta and acceleration coefficients, (3) 20 log Mel-scale filter bank coefficients, their first and second derivatives, zero-crossing rate, short-time energy, four subband energies, spectral centroid, and spectral bandwidth. For low-level feature extraction, a 250-ms segment was further decomposed into frames with a length of 50 ms and 50% overlap. The feature extraction was performed on the frame level. In turn, the feature vector for the entire 250-ms segment was calculated by averaging the per-frame feature vectors.

Moreover, for each low-level feature set, we extracted two GLTE features corresponding to the presence/absence of background noise as they can complement each other [3, 4]. The background noise was subtracted using the minimum statistics estimation and subtraction method [13] when necessary. As a result,

six GLTE feature images were obtained for a scene snippet. Finally, they were concatenated to form a single GLTE feature image $\mathbf{S} \in \mathbb{R}^{F \times T}$, where F is the size of a segment-wise concatenated GLTE feature vector. In particular, to extract GLTE features for training examples, 10-fold cross validation was conducted.

3.2. From CNN to attentive CNN

3.2.1. CNN

We employed the CNN proposed in [6] for template learning and matching. The network architecture is illustrated in Figure 3. The CNN is able to learn templates that are useful for the classification task [4, 3] thank to its over-time convolution and 1-max pooling. These templates are then convolved with an input GLTE image to extract features for classification [4, 3]. Specifically, over-time convolution between a filter $\mathbf{w} \in \mathbb{R}^{F \times w}$ of width w and a GLTE image $\mathbf{S} \in \mathbb{R}^{F \times T}$ resulted in a feature map $\mathbf{o} = (o_1, \ldots, o_{T-w+1}) \in \mathbb{R}^{T-w+1}$:

$$o_i = \mathrm{ReLU}(a_i), \tag{9}$$

$$a_i = (\mathbf{S} * \mathbf{w})_i = \sum_{k,l} (\mathbf{S}[i : i + w - 1] \odot \mathbf{w})_{k,l}. \tag{10}$$

Here, $*$ and $\odot$ denote the convolution operation and element-wise multiplication, respectively. Rectified Linear Units (ReLU) activation [14] is used in (9). We then perform 1-max pooling [15, 16] over time to obtain a most dominant feature which corresponds to the maximum matching score of the template (i.e. the convolution kernel) $\mathbf{w}$ and the input GLTE image $\mathbf{S}$ [4, 3].

Furthermore, since this CNN supports filters with different widths w, using $Q \times R$ filters (i.e. Q sets with different widths w and R filters for each set [4, 3]) will produce a vector of $Q \times R$ features. The resulted feature vector is finally presented to a softmax layer for classification. The network is trained to minimize the cross-entropy loss over the training examples:

$$E(\boldsymbol{\theta}) = -\frac{1}{N} \sum_{n=1}^{N} \mathbf{y}_n \log\left(\hat{\mathbf{y}}_n(\boldsymbol{\theta})\right) + \frac{\lambda}{2} \|\boldsymbol{\theta}\|_2^2, \tag{11}$$

where $\boldsymbol{\theta}$ denotes the network parameters and λ is used to compromise the error term and the ℓ_2-norm regularization term.

3.2.2. Attentive CNN (ACNN)

With the CNN in Figure 3, using R filters of width w will result in a feature map $\mathbf{O} = (\mathbf{o}_1^\mathsf{T}, \ldots, \mathbf{o}_{T-w+1}^\mathsf{T}) \in \mathbb{R}^{R \times (T-w+1)}$ (i.e. a matrix). However, instead of using the 1-max pooling over time as in Section 3.2, we learn an attention weight α_i for a vector $\mathbf{o}_i^\mathsf{T}$ at the time i, where $1 \leq i \leq T - w + 1$, using an attention layer:

$$\alpha_i = \frac{\exp(f(\mathbf{o}_i^\mathsf{T}))}{\sum_{j=1}^{T-w+1} \exp(f(\mathbf{o}_j^\mathsf{T}))}. \tag{12}$$

In (12), $f(\mathbf{o}^\mathsf{T})$ denotes the scoring function of the attention layer which is given by

$$f(\mathbf{o}^\mathsf{T}) = \mathbf{o}^\mathsf{T} \mathbf{W}, \tag{13}$$

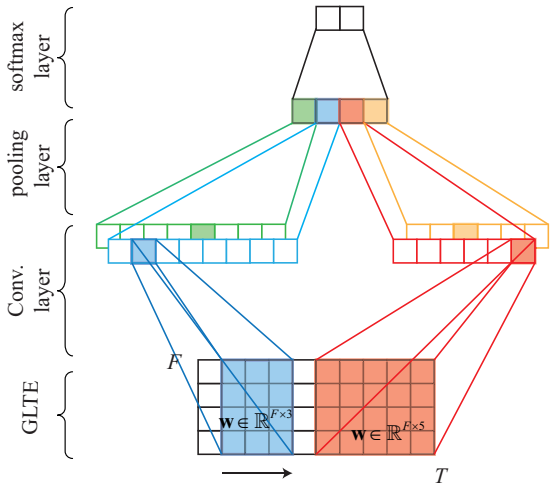

Figure 3: The CNN architecture with convolution over time and 1-max pooling.

where $\mathbf{W}$ is trainable weight matrix of the attention layer. The attentive output feature vector is given by:

$$\mathbf{o}_{\text{att}}^\mathsf{T} = \sum_{i=1}^{T-w+1} \alpha_i \mathbf{o}_i^\mathsf{T}. \tag{14}$$

$\mathbf{o}_{\text{att}}^\mathsf{T}$ is finally presented to a softmax layer for classification as in the case of CNN.

3.3. Data augmentation with GLTE feature

Data augmentation is important to improve generalization of a network [17, 18]. For low-level features, this can be done via systematically synthesizing or transforming existing data. We show that, data augmentation on the high-level GLTE feature can be accomplished by extracting GLTE features for the training data with different data split when cross-validation. We performed cross-validation both with and without taking into account the locations of the audio scenes. All extracted GLTE features extracted were included then into the training set. We experimental saw significant improvements in the classification performance with this data augmentation.

3.4. Calibration with support vector machine (SVM)

As in [3, 5], for all employed networks, after training, we calibrated the final classifier by employing a linear SVM in replacement for the softmax layer for classification. The output vectors of a network extracted from the training examples were used to train the linear SVM classifier which was subsequently employed to classify those output vectors extracted from the test examples.

4. EXPERIMENTS

4.1. DCASE 2017 development data

The "acoustic scene classification" task of the challenge targets 15 categories (cf. Table 1). Each category consists of 312 audio snippets of 10 seconds. The data is split into 4-fold cross validation (cf. [19] for more details).

Table 1: The classification accuracies obtained by different systems.

Category	Development data			Evaluation data				
	Baseline	CNN	ACNN	Baseline	CNN	ACNN	CNN+	ACNN+
Beach	75.3	82.7	81.4	40.7	38.9	41.7	41.7	53.7
Bus	71.8	96.7	95.1	38.9	48.1	45.4	44.4	47.2
Cafe/Restaurant	57.7	68.9	64.4	43.5	61.1	51.9	68.5	64.8
Car	97.1	93.3	91.0	64.8	82.4	79.6	74.1	75.0
City center	90.7	91.0	88.8	79.6	60.2	56.5	57.4	59.3
Forest path	79.5	94.1	94.1	85.2	80.6	67.6	94.4	91.7
Grocery store	58.7	82.4	84.6	49.1	65.7	62.0	66.7	61.1
Home	68.6	76.1	75.2	76.9	73.1	70.4	66.7	70.4
Library	57.1	72.8	73.7	30.6	38.9	35.2	27.8	28.7
Metro station	91.7	89.9	87.9	93.5	85.2	88.9	68.5	75.9
Office	99.7	99.7	99.4	73.1	34.3	33.3	76.9	69.4
Park	70.2	76.3	70.8	32.4	32.4	31.5	21.3	14.8
Residential area	64.1	80.8	78.5	77.8	58.3	52.8	40.7	34.3
Train	58.0	68.5	67.1	72.2	71.3	72.2	71.3	68.5
Tram	81.7	84.2	82.8	57.4	54.6	50.0	54.6	55.6
Overall	74.8	83.8	82.3	61.0	59.0	55.9	58.3	58.0

4.2. Parameters

The classifiers in the tree construction and GLTE extraction algorithms in Section 2 were trained with random-forest classification [20] with 200 trees each. We set the ambiguity threshold in (5) to $\alpha = 0.2$.

The CNN was designed to have three filter sets of widths $\{3, 5, 7\}$ with $Q = 500$ filters each. We set a dropout [21] rate of 0.5 on the output feature vector. The attentive CNN was designed to have a set of 500 filters of width $w = 5$, a dropout rate of 0.2 on the output, and the attention size of 40. Both CNNs were trained for 500 epochs with a batch size of 50.

For all proposed networks, *Adam* optimizer [22] was employed for training with a leanring rate of 10^{-4}. The regularization parameter λ was set to 10^{-3}.

4.3. Experimental results

The classification results obtained by different systems are shown in Table 1. Overall, the CNN achieves the best classification accuracy (83.8%), outperforming the DCASE 2017 baseline over most of the scene categories except for "metro station" on which it slightly underperform the baseline. The average accuracy gains obtained by the CNN is 9.0% absolute over the baseline.

The results obtained by the ACNN are also encouraing, improving the average classification accuracy from 74.8% of the baseline to 82.3%, although is is inferior to that of the CNN with a gape of 1.5% absolute. However, the ACNN shows its efficiency on several categories, such as "forest path", "grocery store", and "library".

In fact, the 1-max pooling over time of the CNN can be interpreted as a special attention mechanism. It fully puts attention on that parts of a scene that best matches to the templates (i.e. the convolutional filters) of the CNN and nulling out the rest. Furthermore, opposed to the typical attention mechanism which is based on time indices (i.e. it learns a common weight for all features at a certain time index), the special mechanism is filter-wise (i.e. it considers each convolutional filter separately). Therefore, features at a certain time index may have different attention weights.

5. THE SUBMISSION SYSTEMS

We developed four systems: CNN, ACNN, CNN+, ACNN+ to submit for Task 1 of the DCASE 2017 challenge. For the former two, their parameters were retained the same as those used for the development data, except for that the whole development data was used for training. For CNN+ and ACNN+, we utilized the GLTE data extracted in the experiments with the development data (both training and test data of all four folds) as additional data augmentation. These GLTE data were extrated with the data splits provided by the challenge (i.e. different subsets of the development data) and, therefore, can reasonably serve for our data augmentation purpose as explained in Section 3.3.

The results of our submission systems are shown in Table 1. Overall, all four systems underperform the DCASE baseline on the evaluation data. It can be explained by the fact that there were some errors in our submission systems related to the cross-validation procedure to extract GLTE features for the training data. As a result, training the networks on these erroneous GLTE features leads to severe overfitting on the evaluation data.

6. CONCLUSIONS

We introduced in this work a novel GLTE representation for audio scene representation. We then proposed two different systems based on CNNs and the attention mechanism and trained them on the GLTE features for audio scene classification. Experiments on the DCASE 2017 development data show significant improvements on the classification accuracy obtained by our systems over the challenge's baseline. An average accuracy gain up to 9.0% absolute is achieved by our systems compared to the baseline. However, the accuracies on the evaluation data are significantly degraded due to errors in our submission systems.

7. REFERENCES

[1] http://www.cs.tut.fi/sgn/arg/dcase2017/.

[2] 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.

[3] 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.

[4] 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.

[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.

[6] 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.

[7] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," *arXiv:1506.07503*, 2015.

[8] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv:1508.04025*, 2015.

[9] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," *arXiv:1502.03044*, 2015.

[10] D. Barchiesi, D. Giannoulis, D. Stowell, and M. Plumbley, "Acoustic scene classification: Classifying environments from the sounds they produce," *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 16–34, 2015.

[11] J. Ye, T. Kobayashi, M. Murakawa, and T. Higuchi, "Acoustic scene classification based on sound textures and events," in *Proc. ACM Multimedia*, 2015, pp. 1291–1294.

[12] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. NIPS*, 2001, pp. 849–856.

[13] R. Martin, "Noise power spectral density estimation based on optimal smoothing and minimum statistics," *IEEE Trans. on Speech and Audio Processing*, vol. 9, no. 5, pp. 504–512, 2001.

[14] 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.

[15] Y. L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proc. ICML*, 2010, pp. 111–118.

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

[17] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Proc. ICDAR*, 2003, pp. 958–962.

[18] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, big, simple neural nets for handwritten digit recognition," *Neural Computation*, vol. 22, no. 12, pp. 3207–3220, 2010.

[19] 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. the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, 2017.

[20] L. Breiman, "Random forest," *Machine Learning*, vol. 45, pp. 5–32, 2001.

[21] 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.

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