End-to-end Audio Classification with Small Datasets – Making It Work

Maximilian Schmitt[1], Björn Schuller[1,2]
[1] *ZD.B Chair of Embedded Intelligence for Health Care and Wellbeing, University of Augsburg, Germany*
[2] *GLAM – Group on Language, Audio & Music, Imperial College London, U. K.*
maximilian.schmitt@informatik.uni-augsburg.de, schuller@ieee.org

***Abstract*—Deep end-to-end learning is a promising approach for many types of audio classification tasks. However, in fields such as health care and medical diagnosis, training data can be scarce, which makes training a neural network from the raw waveform to the target a challenge. In this work, we focus on a public dataset of human snore sounds, categorised into four classes, where one particular class has only a few training samples. We emphasise the pitfalls that need to be taken into account when working with such data and propose an end-to-end model providing a performance similar to that of other deep and non-deep approaches. Furthermore, we show that a model using only convolutional layers outperforms a model employing also recurrent layers.**

***Index Terms*—End-to-end learning, audio classification, representation learning, snore sounds, scarce data**

I. Introduction

Medical datasets designed for training machine learning systems are sometimes limited in regard to their size [1]. This is due to the fact that for rare diseases, only a very limited amount of patients, and therefore samples, are available. Furthermore, hospitals might not have resources to contribute to data collection efforts or face legal constraints. Nevertheless, high accuracies are usually required in the context of medical diagnosis as mistakes can have a big impact on a patient's life. This makes developing a machine learning-based model, where one major paradigm is still 'there is no data like more data' a challenging task. This goes especially for end-to-end models where no hand-crafted features are included in the recognition chain, but a deep neural network is trained in one step from the raw signal to the target.

In this contribution, we deal with the exemplary task of *snore sound* classification, more specifically, the determination of four different types of snoring, depending on the location of obstruction in the throat [2]. The usual procedure for this is to perform a *drug-induced sleep endoscopy* (DISE) with the patient; in contrast to that, an automatic recognition of the snoring type from audio recordings made during the natural sleep of a patient would be fully complication-free.

As shown by Qian et al. [3], hand-crafted acoustic features that are known to be meaningful for certain tasks, e. g., Mel-frequency cepstral coefficients (MFCCs), are far from giving optimal results for the task at hand. End-to-end neural networks have the advantage that the step of feature engineering is avoided, while still being able to extract relevant domain-specific representations of the signal. We will show that also for the employed dataset, deep end-to-end learning from the raw signal yields results comparable to those achieved with hand-crafted features and is robust against changes of hyper-parameters, even though the classes are highly imbalanced.

The following section summarises recent work on end-to-end learning for audio recognition tasks. In Section III, the snore sounds database used throughout the experiments and the corresponding baseline results are introduced. In Section IV, we propose our end-to-end models and motivate architectural choices. Next, experiments and results are presented in Section V. Finally, we conclude and give an outlook on open research questions in Section VI.

II. Related Work

Whereas in image classification tasks, neural networks are usually using the raw pixel values as an input, audio classification is very often relying on hand-crafted features or–at least–spectrogram representations of the audio signal [4], [5]. Using a convolutional neural network (CNN) along the time axis to extract features from the waveform, filters with properties similar to a Mel-filterbank can be learnt autonomously [6]. Nevertheless, these features have been found to perform worse than spectrogram features. Golik et al. found that for automatic speech recognition (ASR) tasks, word error rates obtained with features extracted from trained one-dimensional convolutional layers are only slightly higher than those for a system using the well-established MFCCs [7]. The authors tried lengths from 128 to 1024 for the filter kernel (corresponding to a range of 8 ms to 64 ms for the sampling rate of 16 kHz) without facing a meaningful difference in performance, though they found that a system using two convolutional layers slightly improves the performance. Tüske et al. proposed a model consisting of two convolutional layers, while the 2^{nd} one can be interpreted as an envelope extraction step, using a *rectified linear unit* (ReLU) as activation function for the 1^{st} layer [8]. Though they conclude that the learnt features are less robust when used with another back-end, i.e., classifier model, they perform "nearly equally" to models employing hand-crafted features. Nevertheless, Menne et al. showed that ASR systems using MFCCs are still significantly superior to end-to-end models when training on small amounts of data [9]. Moreover, they also state that learnt features are less robust to noise and mismatches between training and test data pre-processing chains.

978-9-0827-9703-9/19/$31.00

Published works have shown that end-to-end architectures consisting of a CNN using the raw waveform as input and a recurrent neural network (RNN) outperform many other approaches also in the field of speech emotion recognition. Trigeorgis et al. and Tzirakis et al. use two convolutional layers (20 filters of 5 ms length/40 filters of 500 ms length) as a feature extraction front-end [10], [11]. They use ReLU activations to model the rectifying properties of the cochlear transduction in the human inner ear. After the 1^{st} layer, a maximum-pooling across time is applied (pool size 2), after the 2^{nd} layer, a maximum-pooling across channels is performed (pool size 10). Furthermore, they apply *dropout* (factor 50 %) for regularisation. In the back-end, the authors use a two-layer bidirectional *long short-term memory (LSTM)*-RNN to predict time-continuous emotional states. Sang et al. employ a similar architecture for classification of urban sounds [12].

Aytar et al. proposed SOUNDNET, transferring discriminative knowledge from the video domain to the acoustic domain, learning a deep CNN consisting of 8 convolutional layers [13]. Recently, architectures have been proposed that use very small convolutional filter sizes but consequently much more layers. Lee et al. have introduced SAMPLECNN for music classification, a sample-level neural network overcoming the common frame-structure [14]. Their model consists of a stack of nine one-dimensional convolutional layers of a length of only 3 samples each and subsequent maximum-pooling layers, ending up in more than 2 million model parameters for the whole network. The authors report a performance comparable to spectrogram-based CNNs. Pons et al. find that sample-level networks outperform spectrogram-based CNNs for a music tagging task, if enough training data ($\approx$ 1 million samples) are available, while achieving almost the same performance for smaller amounts of data [15]. They use a model similar to that of Lee et al. [14], but with a lower number of filters in each layer (64–256, compared to 128–512 in SAMPLECNN). Pons and Serra further show for a range of music classification tasks, that feature extraction front-ends using random weights perform close to those trained in an end-to-end manner [16]. They conclude that the architecture itself is the most important aspect of the feature extraction part, where sample-level front-ends result in the best performance in their work.

Even though authors in most related works point out the necessity of large amounts of data, in this contribution, we want to show that it is feasible to learn a model from the raw audio signal also from scarce datasets.

III. CORPUS

All experiments in this contribution are conducted on the publicly available Munich-Passau Snore Sound Corpus (MPSSC) [2], introduced in the 2017 *Computational Paralinguistics Challenge* [17]. The dataset comprises 828 audio files containing single snore events from 219 subjects, collected at three different medical centres in Germany. One snore event is an isolated snore sound recorded through a headset during DISE. Each event has been labeled as one out of 4 different snore types, corresponding to the location of the vibration in the upper airways: V (velum), O (oropharyngeal), T (tongue), E (epiglottis). This procedure was done by a trained expert, watching the videos recorded simultaneously with the audio recordings. The snore type is usually constant for all events from the same patient, except for one patient, where 'V' and 'E' type snoring were discovered. Subjects' ages range between 24 and 78 years, with an average age of 49.8 years and no significant difference between snore types; most patients are male. The events have different durations ranging from 0.73 s to 2.75 s (mean: 1.51 s, standard dev.: 0.73 s). All events have been split into three subject-disjunct partitions (Training, Development, Test). Statistics on the number of snore events (i. e., audio files) per class and partition are shown in Table I.

TABLE I
NUMBER OF SNORE EVENTS FOR EACH SNORE TYPE (V, O, T, E) AND EACH PARTITION (TRAINING, DEVELOPMENT, TEST) OF THE MPSSC.

Partition	V	O	T	E	SUM
Training	168	76	8	30	282
Development	161	75	15	32	283
Test	155	65	16	27	263
SUM	484	216	39	89	828

The MPSSC has two main difficulties: Firstly, the number of instances is quite low and the classes are highly imbalanced (only 23 events of class T in Training and Development partition compared to 329 of class V). Secondly, the level and spectrum of the background noise varies across single recordings [2]. The challenge baseline approach uses a large-scale engineered acoustic feature set and a support vector machine (SVM) classifier. As a metric, the unweighted average recall (UAR), i. e., the macro-average recall (mean of the per-class recalls) was used in order to take account of the class imbalance, with a UAR of 58.5 % on the Test partition [17]. The winners of the challenge, Kaya and Karpov, used a fusion of the confidence scores from variants of *extreme learning machine* and *partial least squares regression* classifiers trained on hand-crafted acoustic features and *Fisher vector* representations, achieving a UAR of 64.2 % on the Test set [18]. Competitive results have not only been obtained using deep neural networks but also with standard classifiers such as SVMs [19], [20], and (even) naïve Bayes [3] classifiers, which are known to generalise well also on scarce training data. Amiriparian et al. achieve a UAR of 67.0 % on Test with features computed by a pre-trained image classification network using the spectrogram representations of the snore sounds as an input and an SVM [19]. Qian et al. obtain a UAR of 69.4 % employing wavelet-based bag-of-audio-words [21] representations and naïve Bayes [3], choosing the optimum feature type on the Test set. Demir et al. use a fusion of *histograms of oriented gradients* and *local binary patterns* and an SVM [20], with a UAR of 72.6 % on Test, which is the best result reported so far, choosing the model performing best on the Development set.

Concerning deep learning approaches, Vesperini et al. employ a DNN classifier trained with *Gaussian mixture model* supervectors from the *deep scattering spectrum* space [22].

They achieve a maximum UAR of 67.1 % on the Development set and a UAR of 67.7 % on the Test set with the same model, while achieving an even higher UAR on Test for a model performing worse on the Development set. Wang et al. use a combination of spectrogram-based CNN and *gated recurrent units* (GRU), with a UAR of 63.8 % [23]. Semi-supervised *conditional generative adversarial networks* for this task are proposed by Zhang et al. [24], with a maximum UAR of 67.4 % on the Development set, leading to a UAR of 54.5 % on the Test set.

IV. Methodology

In this section, the processing chain and architecture of the end-to-end neural network model are presented. Most design choices have been made based on the know-how introduced in Section II and/or optimisation on the Training/Development set and are explained in the following subsections.

A. Pre-processing

All files from the MPSSC are provided with a sampling rate of 16 kHz. The waveforms have zero mean and their absolute maximum is normalised to 1.0 for each file separately. In order to deal with the different lengths, signals are continued periodically to match with the longest instance of 2.75 s duration. All classes are *balanced* in the training partition by *upsampling* (duplicating) the instances of the minority classes, to match the number of instances in the majority class. For each training epoch, the instances are *re-ordered* to appear in an alternating order (V, O, T, E, V, O, T, E, ...) so that also each mini-batch is balanced w. r. t. classes. This step prove to slightly increase the stability of the training. As the usual *cross-entropy loss* is weighting each instance with the same weight by default, the balancing is a very meaningful step for the training process.

B. Feature-extraction front-end

On the left side of Fig. 1, the feature extraction front-end architecture is shown. It is mostly inspired by the SampleCNN architectures introduced before, but with some meaningful differences. Preliminary experiments have shown that a lower number of layers and filters captures the signal in a better way. This might be owed to the relatively low amount of training data, the short sequences, and the relatively simple acoustic nature of snore sounds, compared to the music tagging tasks where the architecture was introduced for. We also found that a filter length of 4 (and a stride of 2) is more robust than convolutions with a length of 3.

Starting from the raw unidimensional time signal, we use two blocks consisting of two convolutional layers followed by a maximum-pooling and a dropout layer each. Finally, a single convolutional layer followed by a maximum-pooling and a dropout layer is attached. It is important to note that the dropout is independent for each time step and feature map, unlike *spatial dropout*, where the dropout is constant for the whole input space and varies only across feature maps [25]. All convolutional layers employed are one-dimensional (1D),

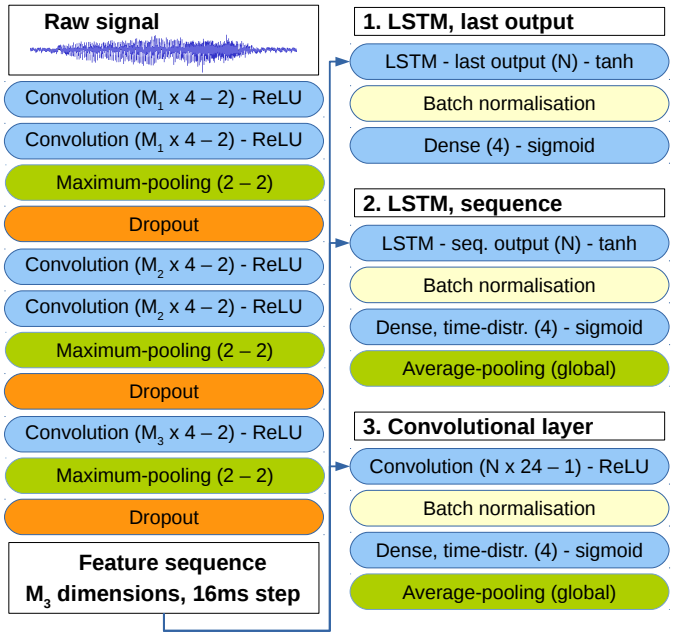

Fig. 1. Left: Feature extraction front-end. Right: Three considered back-end architectures. All convolutional layers are 1D-convolutions with specified (# filters x size – stride). Maximum-pooling is always with pool size and stride 2. Average-pooling is always global.

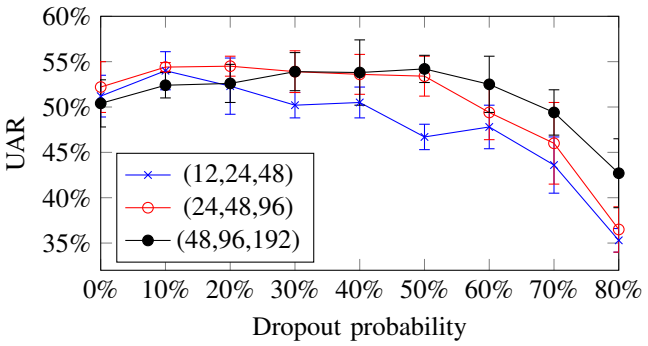

Fig. 2. Experiments to optimise the number of filters and dropout in the feature extraction front-end using the *LSTM, last output* back-end. Results are with a 2-fold CV setup of Training and Development partitions of the MPSSC. Mean and standard deviation over 5 runs are shown.

however, they map between sequences of features, so they are two-dimensional convolutions, where the input in the first layer has only one feature per time step (the waveform). In subsequent layers, all features (outputs) from previous layers are taken into account for each time step. In all convolutional layers, a ReLU activation function is used. It is important to note that, in comparison to SampleCNN, we do not use batch normalisation (BN) in the front-end as we found that it degraded the accuracy. With the proposed front-end architecture and an input sampling rate of 16 kHz, we end up in a feature sequence with a step size of 16 ms, each one taking into account temporal context of an interval of approximately 32 ms. This is comparable to the frame step and size of hand-crafted low-level descriptors [2].

TABLE II
RESULTS WITH DIFFERENT BACK-END ARCHITECTURES. CROSS-VALIDATION (CV) RESULTS ARE THE AVERAGE UAR ON PERMUTED TRAINING AND DEVELOPMENT SETS OF THE MPSSC. ALL RESULTS INCLUDE A FUSION OF MODELS TRAINED FROM 10 DIFFERENT RANDOM SEEDS. FOR THE EVALUATION ON THE TEST SET, PREDICTIONS OF EACH OF THE 10 MODELS TRAINED ON TRAINING/DEVELOPMENT PARTITION ARE FUSED. EXPERIMENTS HAVE BEEN REPEATED 5 TIMES, WHERE MEAN AND STANDARD DEVIATION OF THE UARS ARE REPORTED.

# Units last layer (N)	**UAR CV ± stddev**	**UAR Test ± stddev**
LSTM, last output		
50	55.7 % ± 1.8 %	64.5 % ± 2.1 %
100	56.0 % ± 1.6 %	63.8 % ± 1.7 %
150	55.0 % ± 0.8 %	60.9 % ± 3.0 %
LSTM, sequence output + average-pooling		
50	59.0 % ± 1.0 %	65.1 % ± 0.8 %
100	57.8 % ± 0.8 %	65.9 % ± 1.3 %
150	57.6 % ± 1.2 %	67.1 % ± 0.6 %
Convolutional layer + average-pooling		
50	59.2 % ± 1.2 %	67.8 % ± 0.7 %
100	**60.2 % ± 0.5 %**	**67.0 % ± 1.6 %**
150	59.9 % ± 1.4 %	68.0 % ± 0.9 %

C. Back-ends

For the classification, we compare *three* different back-end models, summarised on the right side of Fig. 1:

1) A bidirectional **LSTM** returning only the **last output**. The output activation is *tanh* as we experienced much better results with it than with ReLU. The LSTM layer is followed by a BN layer and finally a *dense* (i. e., fully-connected) layer with one neuron for each class. The final activation function is a *sigmoid* function as the results were more stable (i. e., independent from the random initialisation) than with *softmax*.

2) A bidirectional **LSTM** returning a **sequence** as output. The output activation is also *tanh* and the layer is followed by BN. The sequence is processed by a *time-distributed dense* layer, i. e., a dense layer at each time-step, with one neuron per class (sigmoid activation). Finally, the sequence is subjected to an **average pooling**.

3) Another **convolutional layer** with an input size of 24, larger than that used in the front-end to exploit a larger temporal context (approximately 400 ms of context). The further layers are exactly the same as in model 2.

The number of LSTM units and convolutional filters is varied in our experiments (see Section V). Attention mechanisms were also tried but did not provide any advantage, which might be due to the comparably short audio signal lengths [26].

V. EXPERIMENTS AND RESULTS

The proposed neural network architecture is implemented in the deep learning framework *Keras*. In initial experiments, a *mini-batch* size of 20 was found to be optimal for all back-ends. *Adam* optimiser is used in the default configuration with *categorical cross-entropy* loss and a learning rate of 5E^{-4}, though model performance was quite stable in a larger range throughout preliminary evaluations. The network is trained for up to 150 epochs, stopping when the maximum on the respective validation partition is reached (see below).

TABLE III
COMPARISON OF RESULTS ON THE MPSSC.

Approach	**UAR Devel.**	**UAR Test**
Baseline MPSSC [2], [17]	40.6 %	58.5 %
Kaya & Karpov [18]	–	64.2 %
Amiriparian et al. [19]	44.8 %	67.0 %
Wang et al. [23]	51.7 %	63.8 %
Demir et al. [20]	37.8 %	72.6 %
Vesperini et al. [22]	67.1 %	67.7 %
Qian et al. [3]	35.0 %	69.4 %
Zhang et al. [24]	67.4 %	54.5 %
Proposed (end-to-end)	59.1 %	67.0 %

In the first round of experiments, we optimise the *number of filters* and the *dropout* rate for the front-end, using the LSTM (last output, 100 units) as a back-end. We consider the average UAR on the Development/Training partition in a 2-fold cross-validation (CV) setup. Three different configurations for the numbers of filters $(M_1, M_2, M_3) = \{(12, 24, 48), (24, 48, 96), (48, 96, 192)\}$ are evaluated, dropout is optimised in the range of $[0\%, 10\%, \ldots, 80\%]$. Each experiment is run 5 times, mean and standard deviations of the UAR are shown in Fig. 2.

In the second round of experiments, in order to have a suitable trade-off between the number of parameters and the model accuracy, we use the feature-extraction front-end with $(24, 48, 96)$ filters and a dropout of 50 %, where the standard deviation between evaluations is low. All back-ends are evaluated, optimising the number of units/filters (N) in the final layers. As the Training and Development sets are relatively small and have almost the same size (cf. Table I), partitions are switched again for validation to obtain results in terms of a 2-fold CV. For the final predictions on the Test set, the model predictions (from the models learnt on Training and Development sets, respectively) are fused by simply multiplying the network outputs for each class. To further increase the robustness, the training process is repeated 10 times and all model predictions are fused. To demonstrate the stability of the final model, each experiment (of 2×10 training cycles) is run 5 times and mean and standard deviations of the UARs are reported in Table II.

It is evident that the convolutional back-end model achieves the best results, independent from the number of units. For a fair evaluation, we consider the configuration obtaining best average results in the Training/Development CV, which is a UAR of 60.2 %, leading to a UAR of 67.0 % on the Test set. An analysis of the predictions shows that the underrepresented classes T and E have recalls of 75.0 % and 85.2 %, respectively, whereas most confusion is between the frequent classes V and O, which is consistent with previous observations [19] and might be due to the spatial proximity of these two classes.

In comparison with other approaches using hand-crafted features (cf. Table III), our approach is keeping up well. Surprisingly, the UAR on the Development set is higher than with most other approaches, which encourages further optimisation of model fusion between Training and Development set. Moreover, a fusion of our Test predictions with

the baseline predictions results in a UAR of 69.8 %, showing that the end-to-end learnt and hand-crafted representations are complementary, which has already been found by Sainath et al. for the task of ASR [27]. Finally, the proposed approach is also computationally quite efficient. On an Nvidia GTX Titan X GPU card, training the proposed optimum architecture with one mini-batch (size 20) takes approximately 63 ms, summing up to a training time of 4:45 minutes when using 150 epochs.

VI. Conclusion and Outlook

We have proven that it is feasible to learn an end-to-end neural network model for audio classification from the raw signal, even with a very low and imbalanced number of training samples at hand. The common finding that dropout–especially in convolutional layers–is often not required when using BN [28] is challenged as we achieved best results with dropout and without BN in these layers. Furthermore, we have shown that for classification on segment level, a convolutional layer followed by a pooling step is superior to an LSTM model.

In the future, it still needs to be proven that the proposed model also works with data from other audio domains, such as speech, body sounds, and environmental noises. Moreover, adversarial training [29] could be a promising extension to further improve the accuracy of the approach.

References

[1] T. Shaikhina and N. A. Khovanova, "Handling limited datasets with neural networks in medical applications: A small-data approach," *Artificial Intelligence in Medicine*, vol. 75, pp. 51–63, 2017.

[2] C. Janott, M. Schmitt, Y. Zhang, K. Qian, V. Pandit, Z. Zhang, C. Heiser, W. Hohenhorst, M. Herzog, W. Hemmert, and B. Schuller, "Snoring classified: The Munich-Passau snore sound corpus," *Computers in Biology and Medicine*, vol. 94, pp. 106–118, 2018.

[3] K. Qian, M. Schmitt, C. Janott, C. Heiser, W. Hohenhorst, M. Herzog, W. Hemmert, and B. Schuller, "A bag of wavelet features for snore sound classification," *Annals of Biomedical Engineering*, vol. 47, pp. 1–12, 2019.

[4] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proc. International Conference on Machine Learning*, Beijing, China, 2014, pp. 1764–1772.

[5] S. Sigtia, E. Benetos, and S. Dixon, "An end-to-end neural network for polyphonic piano music transcription," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 5, pp. 927–939, 2016.

[6] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," in *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Florence, Italy: IEEE, 2014, pp. 6964–6968.

[7] P. Golik, Z. Tüske, R. Schlüter, and H. Ney, "Convolutional neural networks for acoustic modeling of raw time signal in LVCSR," in *Proc. Interspeech*. Dresden, Germany: ISCA, 2015, pp. 26–30.

[8] Z. Tüske, R. Schlüter, and H. Ney, "Acoustic modeling of speech waveform based on multi-resolution, neural network signal processing," in *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Calgary, AB, Canada: IEEE, 2018, pp. 4859–4863.

[9] T. Menne, Z. Tüske, R. Schlüter, and H. Ney, "Learning acoustic features from the raw waveform for automatic speech recognition," in *Proc. DAGA*. Munich, Germany: DEGA, 2018, pp. 1533–1536.

[10] G. Trigeorgis, F. Ringeval, R. Brueckner, E. Marchi, M. A. Nicolaou, B. Schuller, and S. Zafeiriou, "Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network," in *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Shanghai, China: IEEE, 2016, pp. 5200–5204.

[11] P. Tzirakis, J. Zhang, and B. W. Schuller, "End-to-end speech emotion recognition using deep neural networks," in *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Calgary, AB, Canada: IEEE, 2018, pp. 5089–5093.

[12] J. Sang, S. Park, and J. Lee, "Convolutional recurrent neural networks for urban sound classification using raw waveforms," in *Proc. 26^{th} European Signal Processing Conference (EUSIPCO)*. Rome, Italy: IEEE, 2018, pp. 2444–2448.

[13] Y. Aytar, C. Vondrick, and A. Torralba, "SoundNet: Learning sound representations from unlabeled video," in *Advances in Neural Information Processing Systems*, Barcelona, Spain, 2016, pp. 892–900.

[14] J. Lee, J. Park, K. L. Kim, and J. Nam, "SampleCNN: End-to-end deep convolutional neural networks using very small filters for music classification," *Applied Sciences*, vol. 8, no. 150, 2018.

[15] J. Pons, O. Nieto, M. Prockup, E. M. Schmidt, A. F. Ehmann, and X. Serra, "End-to-end learning for music audio tagging at scale," in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018, pp. 637–644.

[16] J. Pons and X. Serra, "Randomly weighted CNNs for $music$ audio classification," in *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Brighton, U. K.: IEEE, 2019, pp. 336–340.

[17] B. Schuller, S. Steidl, A. Batliner, E. Bergelson, J. Krajewski, C. Janott, A. Amatuni, M. Casillas, A. Seidl, M. Soderstrom, A. S. Warlaumont, G. Hidalgo, S. Schnieder, C. Heiser, W. Hohenhorst, M. Herzog, M. Schmitt, K. Qian, Y. Zhang, G. Trigeorgis, P. Tzirakis, and S. Zafeiriou, "The Interspeech 2017 Computational Paralinguistics Challenge: Addressee, cold & snoring," in *Proc. Interspeech*. Stockholm, Sweden: ISCA, 2017, pp. 3442–3446.

[18] H. Kaya and A. A. Karpov, "Introducing weighted kernel classifiers for handling imbalanced paralinguistic corpora: Snoring, addressee and cold," in *Proc. Interspeech*. Stockholm, Sweden: ISCA, 2017, pp. 3527–3531.

[19] S. Amiriparian, M. Gerczuk, S. Ottl, N. Cummins, M. Freitag, S. Pugachevskiy, A. Baird, and B. Schuller, "Snore sound classification using image-based deep spectrum features," in *Proc. Interspeech*. Stockholm, Sweden: ISCA, 2017, pp. 3512–3516.

[20] F. Demir, A. Sengur, N. Cummins, S. Amiriparian, and B. Schuller, "Low level texture features for snore sound discrimination," in *Proc. 39^{th} Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. Honolulu, HI, USA: IEEE, 2017, pp. 3806–3809.

[21] M. Schmitt and B. Schuller, "OpenXBOW: introducing the Passau open-source crossmodal bag-of-words toolkit," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 1–5, 2017.

[22] F. Vesperini, A. Galli, L. Gabrielli, E. Principi, and S. Squartini, "Snore sounds excitation localization by using scattering transform and deep neural networks," in *Proc. International Joint Conference on Neural Networks (IJCNN)*. Rio de Janeiro, Brazil: IEEE, 2018, pp. 1–8.

[23] J. Wang, H. Strömfelt, and B. W. Schuller, "A CNN-GRU approach to capture time-frequency pattern interdependence for snore sound classification," in *Proc. 26^{th} European Signal Processing Conference (EUSIPCO)*. Rome, Italy: IEEE, 2018, pp. 997–1001.

[24] Z. Zhang, J. Han, K. Qian, C. Janott, Y. Guo, and B. Schuller, "Snoregans: Improving automatic snore sound classification with synthesized data," *IEEE Journal of Biomedical and Health Informatics*, pp. 1–11, 2019.

[25] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, "Efficient object localization using convolutional networks," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*. Boston, MA, USA: IEEE, 2015, pp. 648–656.

[26] Z. Ren, Q. Kong, K. Qian, M. D. Plumbley, and B. Schuller, "Attention-based convolutional neural networks for acoustic scene classification," in *Proc. 3^{rd} Workshop on Detection and Classification of Acoustic Scenes and Events*, Surrey, U. K., 2018, pp. 39–43.

[27] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals, "Learning the speech front-end with raw waveform CLDNNs," in *Proc. Interspeech*. Dresden, Germany: ISCA, 2015, pp. 1–5.

[28] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. International Conference on Machine Learning*, Lille, France, 2015, pp. 448–456.

[29] J. Deng, N. Cummins, M. Schmitt, K. Qian, F. Ringeval, and B. Schuller, "Speech-based diagnosis of autism spectrum condition by generative adversarial network representations," in *Proc. 7^{th} International Digital Health Conference*. London, U. K.: ACM, 2017, pp. 53–57.