Robustness of Acoustic Scene Classification Using a CNN in a Real-World Scenario

Bjarke B. Madsen, Gergely Kiss, Magnus Borresen and Michail Kampitakis
Department of Electronic Systems, Signal Processing, Aalborg University
Frederik Bajers Vej 7, Group 770, 9220, Aalborg, Denmark
Email: [bmadse18, gkiss21, mborre15, mkampi21]@student.aau.dk

***Abstract*—In this paper, we present a convolutional neural network (CNN) data-driven approach for acoustic scene classification. The aim of this paper is the development of an audio CNN classifier and test its robustness when the original signal is imposed on noise. Spectrogram images are created from the signal which are used as an input for the CNN classifier. Test data was imposed with environmental noise (speech and wind) and simulated channel noise (Gaussian and packet loss) added to the original audio signal, simulating a non-ideal environment. The accuracy of the initial model was 95% and 47% with original and average with signal imposed on noise respectively.**

1. Introduction

Determining the surroundings in which an audio signal was recorded can be a challenging task. Nevertheless, the information can be useful for more intelligent sensing technology. For example, acoustic monitoring can be used to identify acoustic event or acoustic environments. The former is relatively short physical event, like breaking glass, while the latter is considered as identification of longer recordings, for example of how windy an environment is, or how heavy the traffic is on a road [1]. Audio analysis of acoustic environments can be used for acoustic scene classification (ASC). ASC aims to associate an audio recording with the best fit from a predefined set of acoustic scenes [1]. Workplace environments such as construction sites, freight terminals, airports etc. can generally be defined as having a finite amount of specific acoustic scenes. In an airport, scenes could include: The terminal, luggage area, next to the plane, in the office amongst others. The same type of scenes can be identified for other work environments as well.

In this paper we investigate a case of high-noise work environments that has a variety of acoustic scenes. The results of an ASC algorithm can be used as a supplement for adaptive signal processing algorithms, such as noise reduction, location tracking etc. For several years ASC has been the driving factor behind the detection and classification of acoustic scenes and events (DCASE). DCASE holds annual challenges which over the years has shown that ASC using different techniques, such as neural networks, are possible with descent accuracy for a variety of scenes as result [2]. The datasets used for the challenges usually consists of 10-15 urban acoustic environments. These environments are suitable for scientific research, but not necessarily representative of a practical use case for ASC algorithms. For the past few years papers from the DCASE use model-based approaches to solve the classification problems. In a model-based approach, predefined acoustic features are extracted from the data and used in combination with neural networks such as shown in [3], [4], [5]. These features may be beneficial for the sake of the recordings from the DCASE competition, however they may not be for the general and practical case. More data-driven approaches have also been investigated in DCASE, where no specific features are extracted and used as input. Instead, the raw audio data or a spectral representation of the raw data is used as the input for neural networks. This is seen in [6], where a classification algorithm with an input of windowed linear-frequency (spectrogram), a raw waveform and an i-vector representation of the signal is proposed. Though there are many classification structures and algorithms that can be used for ASC, a review of neural network ASC methods in [7], concludes that CNN is the most utilized approach amongst neural networks in ASC.

The idea behind the data-driven approach is to make it convenient to add more training data to a model or be able to generate a new model classifying a different environment based on parameters from the previous model. Therefore this paper tries to show that a simple data-driven CNN model can perform well in a practical scenario. To emphasize this, a dataset will be collected as well by combining self-made recordings with online data sources.

In post-processing audio is assumed to be recorded on a headset which is connected through a wireless channel to a base station, on which the classification is done. Therefore signals could be subjected to multiple types of noise. From the surrounding environment, wind noise and speech could influence the classification accuracy. The electronic system itself could add white Gaussian noise (WGN). Wireless channels can be unstable and packet loss can occur. In this paper we investigate the robustness of the before mentioned simple CNN implementation when signals are subjected to the noise types described above. For the implementation, Python and the Deep-learning library Keras will be used.

The paper is organized as follows: In section 2-A, the training data is acquired and preprocessed. The CNN baseline model is described in section 2-B. A noise module that can impose different types of noise to test data is described in section 2-C. Results are assessed in section 3, and the final findings in this paper is summarized in sections 4 and 5.

2. Materials and Methods

A. Data Acquisition and Preprocessing

An acoustic scene dataset was recorded at Nyt Aalborg Universitetshospital (NAU) [8] construction site. To obtain a diverse dataset of recordings, multiple microphones were utilized with different responses to capture as much audio information as possible. A *Zoom H4n Handy Recorder* was used to record with coincident X-Y spaced microphones at 90° for wide image capturing of space. The recorder supports a four-channel mode, that allows to record with two additional external microphones, where *PreSonus PRM1* and *t.bone MM-1* were recorded on separate channels. This allows capturing of more data at the same time, as well as creates diversity on the recordings due to different distances between the microphones. The external microphones were placed in a distance of around 2 meters from the recorder during the recording process for wider area of coverage. The sampling rate of the recording was 44.1 kHz. The resolution of the sampling was 24 bit, which was converted to 32-bit floating-point numbers in the recorder.

The recordings from NAU are sorted by their acoustic scene and added to the dataset shown in table I: Inside enclosed buildings that is still under construction *(Inside)*, inside a vehicle *(Inside vehicle)*, inside office buildings *(Office)*, outdoors *(Outside)*, and inside a building under construction without any external walls *(Semi-outside)*. This scene division was made due to high variety of natural sounds, which makes the different scenes distinguishable.

TABLE I
Duration in seconds of each class in the dataset, recorded from NAU

Acoustic scene	**Duration (sec)**
Inside	2186.9
Inside vehicle	194.9
Office	2124.5
Outside	2119.5
Semi-outside	1293.6
Total	**7919.4**

A visualisation of the general preprocessing chain can be seen in figure 1.

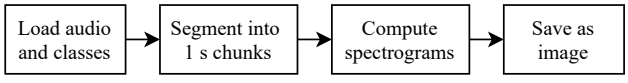

Fig. 1. Preprocessing chain for collected acoustic scene training dataset.

Audio files with multiple channels are separated into a list of two one-dimensional arrays, for processing simplicity. Segmenting the audio recordings into smaller chunks makes the simulation closer to the practical use case. The smaller chunk size simulates a more frequent classification update, which may be closer to a practical example of e.g. use cases on a microcontroller.

After segmentation into chunks of 1 second, the dataset is split into following categories: training (81%), validation (10%), and test (9%) data. Every chunk is converted into a linear-frequency spectrogram with SciPy Python library [9]. Spectrograms are computed with a Hanning-window [1] of 1024 samples and an overlap of 512 samples.

Saving the spectrograms as PNG images makes visual evaluation easier, but also augments the spectrograms to an 8-bit unsigned integer 85×513 RGB image. As spectrograms are a colour scale representation of spectral energy, the data is automatically further augmented into three different channels (RGB). The colour representation can potentially change the accuracy of the CNN, however this topic is not studied in this paper.

B. CNN Model

After preprocessing the data, the spectral images become the input of the convolutional neural network. The CNN model was done in Keras, which is a high-level application programming interface (API) for neural networks written in Python [10], [11].

The input of the neural network consists of 54644 spectral images generated from 2.2 hours of recorded audio, supplemented with online data. During training, the spectral images are loaded in smaller batches. The distribution of the spectral images are shown in table II.

TABLE II
Number of spectral images used in training and prediction process

Class	Training	Validation	Test
Inside	13553	1660	1463
Inside vehicle	4248	514	458
Office	8211	956	949
Outside	9045	1194	999
Semi-outside	9774	841	779

During training, the images are shuffled in the batches, so the outcome of the training does not rely on any specific order of the images. When testing the neural network, no shuffling is applied on the images, because the unshuffled order of the images is needed for generating the confusion matrix. The confusion matrix is used to visualize the true and false predictions from the test dataset [12].

The implemented neural network is based on a structure commonly used for image classification. The structure of the implemented CNN is depicted in figure 2. The input of the convolutional network is a three-dimensional Keras tensor, of the input image. The input layer is followed by a convolution layer, which is responsible for the feature extraction [12]. In the first convolutional layer, 32 kernels were used with the size of 3×3. Considering the resolution of the input image and previous experiments, the mentioned size and number of the kernels were found to be the most effective for this classification problem. A padding function follows the convolutional layers, so the output size is the same as the input size by padding it with zeros. The activation function after the convolution is a rectified linear unit (ReLu) activation function

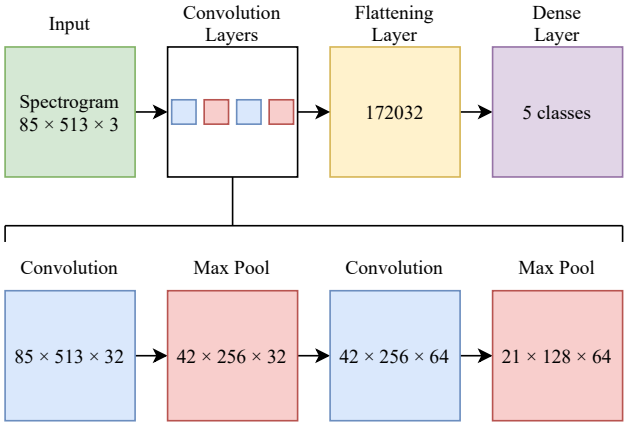

Fig. 2. The flow chart of the CNN implemented for acoustic scene classification. The dimension of the output tensors are shown in the blocks.

[10]. After the convolutional layer a max pool layer [10] decreases the image size to its half, reducing the computational cost. The second convolutional layer is similar to the previous with identical kernel size, activation function, and padding, but with 64 kernels instead of 32. The second max pool layer is identical to the previous one. The flattening layer converts the tensors into a one-dimensional array in a structured manner, before passing it to the next layer which is a dense layer [11]. The amount of output nodes in the dense layer corresponds to the number of classes. Outputs are a probability distribution of classes from the softmax activation [10]. The output prediction from the network is the class with the highest probability.

After describing the model structure, the next task is describing the loss function and the optimizer for the training process. Since this neural network model is built for a multi-class single-label classification problem, the loss function is categorical cross entropy [10]. The loss function determines the error of predictions calculated at the end of each training iteration. This loss function should be minimized, to minimize the training error. Stochastic gradient descent (SGD) and Adam are used as the most common optmizers in the DCASE [2]. Adam has become more popular, due to an adaptive learning rate [10]. If the learning rate is too low, it will require a high number of updates before reaching the minimum. A too large learning rate can lead to divergent behaviour. The optimal learning rate will quickly reach the minimum, which was found by experimentation to be 0.0001.

The model is trained in several iterations called epochs. In each epoch the whole dataset runs through the model. The selected number of epochs was chosen after experimentation. With each epoch, the training and validation accuracy should increase, which means that if the amount of epochs is too low, the validation and test accuracy will be low too. However if the model is trained with too many epochs then over-fitting will happen, which means that the model is very accurate classifying the training data, but it is unable to properly predict validation and test data. The implemented CNN was trained with 5 and 10 epochs.

C. Noise

In order to check the robustness of the model in a non-ideal environment, test processes were implemented with various types of signal interferes which could reduce the accuracy of the existing model. These processes were done in the time-domain, to more realistically mimic real-word scenarios where the signal would suffer from some type of noise before the analysis and spectrogram extraction processes.

In addition, by adding noise to signal, the original data can be used for both testing with and without noise, resulting to a fair comparison.

1) Speech noise: Since the model is intended to work with headsets, human voice (such as communication through the headset or people talking nearby) can also be recorded with the headset's microphone, leading to signal interference. In addition, human voice is not necessarily a defining feature in the acoustic scenes, and for that reason it can be considered as unwanted noise.

To evaluate the effect of the voice on classification process, a speech signal from the open-source TIMIT database [13] was added to the existing test data. The speech signals were chosen randomly as well as their positioning in the test data. The added speech samples do not vary in speaker loudness due to the nature of the TIMIT dataset, resulting to being more audible in quiet audio and less in noisier environment. By repeating the process, the density of the human voice imposed on the original signal can be increased, making the classification more challenging for the model.

2) Wind noise: Blowing wind on a microphone capsule produces non-periodic, non-stationary noise, which can alter the quality of the recorded signal and therefore, the acoustic scene features. It also makes the different acoustic scenes more correlated to each other, which is a highly undesired condition in acoustic scene classification. The wind noise is only added in outside based classes (namely in *Inside vehicle*, *Outside* and *Semi-outside*) where it could occur in practice.

Wind noise can be either recorded or simulated before adding it to the desired test samples. Therefore, open-source wind audio samples were added onto the test samples of our database [14]. For every test sample, a random starting point in the wind samples was chosen to introduce more variety for multiple test samples, and the corresponding samples after this point are added to the chunk file signal.

Wind noise is nonlinear, meaning that in a practical scenario, a windy acoustic scene is not a linear combination of a clean audio and wind noise. However, modeling windy environment by adding randomly picked wind sound to the original data gives audibly very close results to the practical example.

3) System and Channel noise: Unlike the previously explained noise types, system and channel noise can be induced during a wireless transmission of the signal. Two main types

of noise are the WGN and the channel packet loss. The way they are added to the signal is depicted in figure 3.

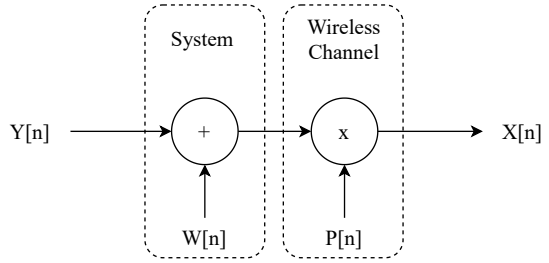

Fig. 3. Model of system and wireless channel with noise and packet loss processes. $Y[n]$ is the input signal, $X[n]$ the output signal, with $W[n]$ AWGN and $P[n]$ packet loss

Additive white Gaussian noise (AWGN) is a way to mimic noise imposed on real world random processes (signals). The noise samples are drawn from and independent and identical zero-mean normal (Gaussian) process $W[n] \sim \mathcal{N}(0, \sigma^2)$ and are added to the signal. The AWGN is used because it has a uniform power distribution across the frequency spectrum. The noise $W[n]$ is assumed to be uncorrelated with the signal $Y[n]$. The implemented function to add AWGN makes it possible to vary the variance, such that different values can be imposed on the test data.

In wireless communication systems, packet loss during signal transmission has a certain probability of occurring and as a result, important information may be lost. Assuming that the classification is operated remotely, it is desired to simulate this phenomenon and to do so, two methods of packet loss were implemented, random loss and burst loss. The former simulates the sparsely loss of information while the latter focused and continuous loss [15]. Both of these functions can be set with specific percentage of p_{loss}. The simulation is made by splitting chunk files into simulated packets $Y[n] = \{y_0, y_1, ...y_{n-1}\}$ before it is further processed. These y packets consist of $sr \cdot t$ audio samples, where sr the source sample rate and t the time of the transmitted packet in seconds. In this implementation, 0.1 seconds is used for packet size, since it is considered to be a realistic number. For each x packet, a corresponding $P[n] = \{p_0, p_1, ...p_{n-1}\}$ value indicates the state of the packet. The samples p_i takes values either 1 or 0 with the latter meaning that the packet is lost.

Random packet loss consists of a Bernoulli random process with length equal to n [15]. The probability of successfully arrived packets in the process is $1 - p_{loss}$ while the rest are lost and the output of this process is:

$$X[n] = Y[n] \cdot P[n] + W[n] \tag{1}$$

Which is depicted in figure 3. A spectrogram from *Outside* class showing 5% random packet loss is represented in figure 4. The illustrated spectrogram image is 10 seconds, so the results could be more visible.

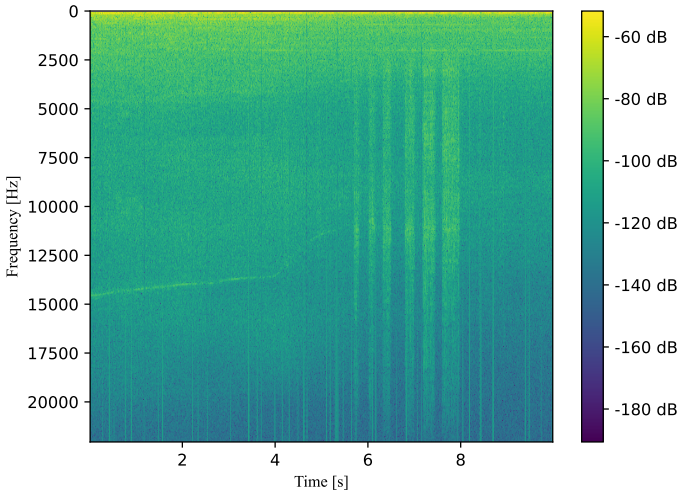

Fig. 4. Spectrogram of *Outside* with 5% random packet loss. y-axis illustrates the frequency in hertz and x-axis the time in seconds. The thin vertical lines illustrate the lost packets.

In burst packet loss, both the amount of packets loss in each burst and the burst position should be random. As an example, The burst packet loss output is shown in figure 5.

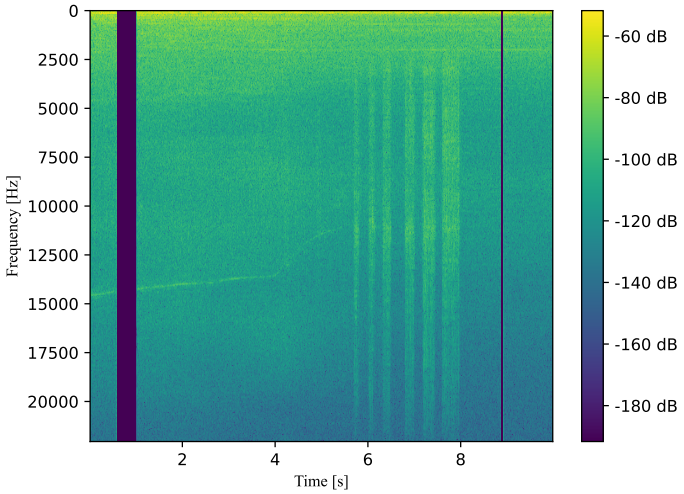

Fig. 5. Spectrogram of *Outside* with 5% burst packet loss. y-axis illustrates the frequency in hertz and x-axis the time in seconds. The thick vertical lines illustrate the lost packets.

A simple method of generating loss of neighbouring packets has been implemented. First, the amount of total packet loss, which for comparison reasons is also 5%, is computed and then a random choice determines if one or two bursts should appear in a chunk. If a single burst occurs, the burst around this point contains all the lost packets. For dual bursts, the distribution of packets lost in each burst is created from the uniform random variable $\mathcal{U}_1(0, 1)$. Meaning $B_1 \sim U_1(0, 1)$ and $B_2 = 1 - B_1$. Where B_1 is the percentage of packages lost in the first burst and B_2 is the percentage of packets lost

in the second burst. The probability of a single burst is set to ¾. Dual bursts will therefore occur with a probability of ¼. The output $X[n]$ of the burst process is computed as shown in equation 1. I WiFi-networks burst loss is more likely to occur [15] and therefore it assumed to be a more realistic model in this system as well to test the robustness of the model.

3. Results

The network was tested with the noise types described in section 2-C with two different number of epoch. The results are shown in table III. System and channel noise is abbreviated as **S&C**.

TABLE III
CLASSIFIER ACCURACY IN PERCENTAGE WITH DIFFERENT NOISE TYPES. RESULTS WITH RANDOM PACKET LOSS IS SHOWN IN PARENTHESES. ALL INCLUDES EVERY NOISE TYPE.

Model	Original	S&C	Wind	Speech	All
5 epochs	94%	67% (89%)	74%	46%	42% (44%)
10 epochs	95%	70% (92%)	78%	49%	44% (47%)

The results shows that the model performs fairly well in conditions where packet loss is random. Burst packet loss and wind noise does decrease accuracy. However predictions should are still usable.

The results also shows that speech noise is generally the worst noise type for signals to be imposed to.

Table III also shows that increasing the number of epochs does not necessarily increase the accuracy. Although when trained with 10 epochs, the ratio of the true and false predictions is higher when no noise was added, it is also visible that the prediction accuracy decreases when noise is added.

Figure 6 shows how added noise alters the true and false predictions. Due to the differentiation of the amount of test samples, the confusion matrix is normalized.

AWGN barely changes the overall accuracy. There are very high numbers in the diagonal, while there were barely any false predictions resulting 0 or very low numbers in the other indices.

Packet loss is added to all images and is therefore representing the absolute worst case scenario. In a real-world scenario, packet loss would be expected to be significantly reduced as many communication systems might drop the link, when packages are lost for a prolonged period of time. Burst packet loss is mainly misclassified as the *Inside* class. This may be due to the fact that the class contains recordings of a very quite empty building.

Speech is generally only represented in the *Office* class. This becomes apparent looking at figure 6, since most false predictions with speech noise is in the *Office* class. The issue could be worsened by for example if multiple people were speaking in the near vicinity of the microphone. It could also be alleviated by adding more recordings with speech in the individual classes.

Wind noise also decreased the overall accuracy. As seen in figure 6, most predictions were misclassified as *Inside Vehicle*,

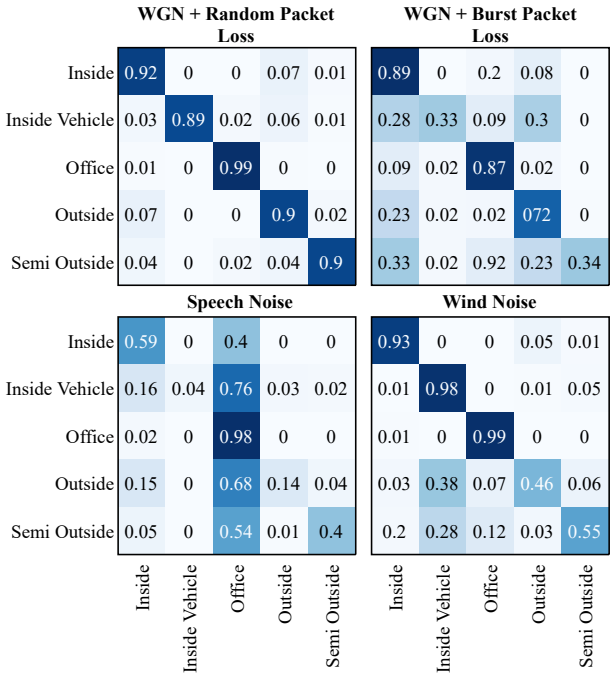

Fig. 6. Test Confusion Matrix for 5 epochs. Columns represent the predictions, while the rows represent the true class of the data.

which likely means that wind noise is a strong feature of the before mentioned class. Since the class contain a recording of a golf-car like vehicle driving, there is a large amount of wind noise.

4. Discussion

The trained model shows promising results in predicting the acoustic scenes. AWGN does not seem to have any effect on predictions, while burst packet loss and wind noise makes the prediction accuracy decrease and speech noise does make the predictions unusable. The acoustic scene where the majority of speech in the recordings were *Office*. This observation may explain why most scenes are predicted incorrectly to be *Office* when imposed to speech noise.

The objective of the paper has not been to accurately model packet loss in a wireless communication system. If packet loss were to be correctly modelled, Bernoulli random sequences should only be used for comparison and Markov chain state machines should be used as the primary packet loss generator instead [15].

As the results show, the model seems to have found features that resembles the characteristics of the scenes. Therefore, the training data does create limitations on the prediction accuracy when introducing noise. Since only the *Office* scenes had recordings of human speech, it naturally puts a constraint on the model, when trying to add a speech to all scenes. If wind noise was added to all scenes, most scenes were classified as scenes that had wind noise in the training data. The method of adding wind noise is debatable since wind noise is not a additive noise type. To correctly represent the effect of wind

noise, it should be present at the time of recording. Likewise, tools such as drills, jack hammers etc. seems to show up as features as well, as scenes are often miss-classified with these noise types introduced. To alleviate this issue, more diverse training data should be gathered.

5. Conclusion

In this paper we presented a method for acoustic scene classification using CNN. The CNN model had high classification accuracy when no noise was imposed on the acoustic field. In noisy environment the accuracy decreased, however it still seems to be usable for practical case. Further data augmentation and more training data can further improve the model accuracy. Further testing of the CNN structure is needed in order to determine whether the model can be used for other environments.

Acknowledgments

We thank RTX for the project proposal and supplementary project supervision. We also thank Jorton A/S for access to the construction site NAU, and Thomas Vestergaard for a private donation of equipment for this purpose.

References

[1] Tuomas Virtanen, Mark D. Plumbley, and Dan Ellis. *Computational Analysis of Sound Scenes and Events*. 1st. Springer Publishing Company, Incorporated, 2017. ISBN: 3319634496.

[2] DCASE Community. *Detection and Classification of Acoustic Scenes and Events*. 2013. URL: http://dcase.community/events (visited on 11/16/2021).

[3] Hu Hu et al. *Device-Robust Acoustic Scene Classification Based on Two-Stage Categorization and Data Augmentation*. Tech. rep. DCASE2020 Challenge, June 2020.

[4] Wei Gao and Mark McDonnell. *Acoustic Scene Classification Using Deep Residual Networks with Focal Loss and Mild Domain Adaptation*. Tech. rep. DCASE2020 Challenge, June 2020.

[5] Sangwon Suh et al. *Designing Acoustic Scene Classification Models with CNN Variants*. Tech. rep. DCASE2020 Challenge, June 2020.

[6] Jee-weon Jung et al. *DNN Based Multi-Level Features Ensemble for Acoustic Scene Classification*. Tech. rep. DCASE2018 Challenge, Sept. 2018.

[7] Jakob Abeßer. "A review of deep learning based methods for acoustic scene classification". eng. In: *Applied sciences* 10.6 (2020), pp. 2020–. ISSN: 2076-3417.

[8] Region Nordjylland. *Projektet NAU*. URL: https://nau.rn.dk/ (visited on 11/23/2021).

[9] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[10] Francois Chollet. *Deep Learning with Python*. Manning Publications, 2017. ISBN: 9781617294433.

[11] *Keras PI reference*. 2021. URL: https://keras.io/api/ (visited on 12/02/2021).

[12] Tuomas Virtanen, Mark D. Plumbley, and Dan Ellis. *Computational Analysis of Sound Scenes and Events*. 1st. Springer Publishing Company, Incorporated, 2017. ISBN: 3319634496.

[13] Michael Fekadu. *DARPA TIMIT Acoustic-Phonetic Continuous Speech*. URL: https://www.kaggle.com/mfekadu/darpa-timit-acousticphonetic-continuous-speech?select=data (visited on 11/19/2021).

[14] Institute for Communication Systems. *Wind Noise Database*. URL: https://www.iks.rwth-aachen.de/en/research/tools-downloads/databases/wind-noise-database/ (visited on 11/20/2021).

[15] Carlos Alexandre Gouvea Da Silva and Carlos Marcelo Pedroso. "MAC-Layer Packet Loss Models for Wi-Fi Networks: A Survey". In: *IEEE Access* 7 (2019), pp. 180512–180531. DOI: 10.1109/ACCESS.2019.2958260.