Stefan C. Kuhs, BSc

Where am I?
Acoustic Location Classification with Temporal Lags

Master's Thesis

to achieve the university degree of

Master of Science

Master's degree programme: Computer Science

submitted to

Graz University of Technology

Supervisor

Dipl.-Ing. Dr.techn. Roman Kern

Institute for Interactive Systems and Data Science
Head: Univ.-Prof. Dipl.-Inf. Dr. Stefanie Lindstaedt

Graz, March 2020

Affidavit

Eidesstattliche Erklärung

Datum	Unterschrift

Abstract

The artificial classification of audio samples to an abstraction of the recorded location (e.g., *Park*, *Public Square*, etc.), denoted as Acoustic Scene Classification (ASC), represents an active field of research, popularized, inter alia, as part of the Detection and Classification of Acoustic Scenes and Events (DCASE) challenge. Nevertheless, we are more concerned to artificially assign audio samples directly to the location of origin, i.e., to the location where the recording of the corresponding audio sample is conducted, which we denote as Acoustic Location Classification (ALC). The evidence for the feasibility of ALC contributes a supplementary challenge for acoustics-based Artificial Intelligence (AI), and enhances the capabilities of location dependent applications in terms of context-aware computing. Thus, we established a client-server infrastructure with an Android application as recording solution, and proposed a dataset which provides audio samples recorded at different locations on multiple consecutive dates. Based on this dataset, and on the dataset proposed for the DCASE 2019 ASC challenge, we evaluated the application of ALC, along with ASC, providing a special focus on constraining training and test sets temporally, and locally, respectively, to ensure reasonable generalization estimates with respect to the underlying Convolutional Neural Network (CNN). As indicated by our outcomes, employing ALC constitutes a comprehensive challenge, resulting in decent classification estimates, and hence motivates further research. However, increasing the number of samples within the proposed dataset, thus, providing daily recordings over a comparatively long period of time, e.g., several weeks or months, seems necessary to investigate the practicality and limitations of ALC to a sufficient degree.

Acknowledgements

Kern, Roman

Harzl, Annemarie

Struger, Patrick

Neuhold, Sarah

Öhninger, Michael

Rehbichler, Christoph

Wieland, Alex

Butulla, Dominik

Family & Friends

Thank You.

Graz, March 2020 Stefan Kuhs

Contents

List of Figures

Acronyms

Adam Adaptive Moments. 3, 5, 12–14, 44
AI Artificial Intelligence. iv
ALC Acoustic Location Classification. iv, 1, 2, 4, 16, 17, 19, 24, 25, 32–34, 39, 40, 48–50, 66, 81, 82, 84–86, 91–93, 96–98
API Application Programming Interface. 31
ASC Acoustic Scene Classification. iv, 1–4, 11, 16, 17, 19, 24, 25, 32–34, 39, 40, 44, 48–51, 54, 56, 59, 60, 62, 65, 66, 68, 69, 71–73, 76, 77, 80, 92, 96–98

BGD Batch Gradient Descent. 12, 13
BN Batch Normalization. 3–5, 9, 10

CNN Convolutional Neural Network. iv, 2–7, 12, 13, 16, 17, 24, 25, 32, 39, 40, 44, 48, 50, 56, 64, 66, 85, 96
CV Cross-validation. 3–5, 71–73, 84, 85, 91–93, 97

DCASE Detection and Classification of Acoustic Scenes and Events. iv, 1–4, 12, 33, 34, 40, 44, 48–51, 54, 56, 96, 98
DGD Deterministic Gradient Descent. 13
DNS Domain Name System. 17

ERD Entity Relationship Diagram. 33
ES Early Stopping. 2, 3, 5, 11, 39, 44, 48, 50, 51, 54, 59, 66, 68, 81

FFNN Feedforward Neural Network. 3–7

GMM Gaussian Mixture Model. 4
GPS Global Positioning System. 31

ICS Internal Covariate Shift. 9, 10

JWT JSON Web Token. 32

Acronyms

K-fold CV K-fold Cross-validation. 10

MBGD Mini-batch Gradient Descent. 12–14
MBSGD Mini-batch Stochastic Gradient Descent. 13
MFCCs Mel-frequency Cepstral Coefficients. 4
ML Machine Learning. 5, 10
MLP Multilayer Perceptron. 5

NN Neural Network. 2, 9–11, 51

PCM Pulse-code Modulation. 23, 30–32
PoC Proof of Concept. 48, 50, 56, 57

RAM Random Access Memory. 30
ReLU Rectified Linear Unit. 5, 44
REST Representational State Transfer. 16, 17

SDK Software Development Kit. 17
SELD Sound Event Localization and Detection. 98
SGD Stochastic Gradient Descent. 13
STFT Short-time Fourier Transform. 39, 41

TAU development dataset TAU Urban Acoustic Scenes 2019, Development dataset. 1, 2, 11, 16, 33, 34, 38, 39, 41, 44, 48–51, 56, 59, 60, 62, 64–66, 69, 72, 82, 92, 96
TAU evaluation dataset TAU Urban Acoustic Scenes 2019, Evaluation dataset. 48, 56
TAU mobile dataset TAU Urban Acoustic Scenes 2019 Mobile dataset. 41
TL Transfer Learning. 98

WAV Waveform Audio File Format. 31

1. Introduction

Especially due to the DCASE Challenge[1], ASC, the task of artificially assigning audio samples to an abstract class which represents the acoustic environment of the location where the recording is conducted, receives a high amount of attention in recent research.
Although the classification of audio samples to an abstraction of the recorded location represents an active research area in computer science, we are more concerned with the classification of audio samples directly to geographic locations. Therefore, the objective of this thesis covers the investigation of classifying audio samples to the location where the corresponding recording is conducted. We denote this classification task of assigning audio samples to the location of origin as ALC.
More specifically, we examine the capabilities and limitations of applying ALC on a specifically for this evaluation established dataset. Furthermore, the relationship between ALC and ASC, w.r.t. common properties and restrictions, is investigated. In addition, we evaluate the TAU Urban Acoustic Scenes 2019, Development dataset (TAU development dataset), proposed for the DCASE 2019 ASC challenge, for the application of ALC, and analyze the outcomes in consideration of our proposed dataset.
First, we establish a client-server based solution for the process of collecting audio recordings of multiple locations. To provide our solution to a broad audience, the clients are realized by an Android application, hence, smartphones are employed as recording devices. Utilizing a dedicated software solution preserves consistent audio properties for the recordings, and enforces the collection of the necessary metadata. Thus, the monophonic audio samples are recorded with a sample rate of 48 kHz, and a bit depth of 16 bit, utilizing the *UNPROCESSED*[2]audio source.
In terms of metadata, each recording is assigned to a unique location which belongs

[1]`http://dcase.community/` (Accessed on: 2019-10-24)
[2]`https://developer.android.com/reference/android/media/MediaRecorder.AudioSource.html#UNPROCESSED` (Accessed on: 2019-10-24)

to an acoustic scene. Furthermore, multiple events, e.g., ringing church balls, may be added to the particular recording. The recording device, i.e., the smartphone model, is stored for each recording as well. Additionally, to offer feedback on the expressiveness of the published audio recordings, the mobile application features the evaluation of ASC and ALC, based on a server-side employed CNN.
Due to the application of CNNs in multiple state-of-the-art classification benchmarks, e.g., on ImageNet [Deng2009], we evaluate the performance of the acquired dataset, and of the TAU development dataset, on a 2-layer deep CNN, utilizing 40-band log-mel spectrograms, with a window size of 40 ms and an overlap of 20 ms, of 10 s audio samples as features. More specifically, we employ the suggested baseline for the DCASE 2019 ASC challenge, proposed in [Mesaros2018], additionally utilizing a separated validation set and an enhanced Early Stopping (ES) methodology by stopping the training after a predefined number of stagnating epochs w.r.t. the validation loss.
To provide a meaningful evaluation for the application of ALC, we collect audio recordings at 8 locations within the city of Graz (AUT). The locations are equally distributed between the acoustic scenes *Urban Green Space* and *Public Square*. For each location, we record 5 minutes of audio data on each of 3 consecutive working days between 9:00 am and 12:00 am.
Thus, a special focus of this work lies in the evaluation of temporally constrained train-test splits for ALC. To the best of our knowledge, this is the first work evaluating temporally constrained ALC in particular.
The remainder of this thesis is organized as follows. The first part, Section 2.1, of chapter 2, introduces recent research in the field of artificial audio classification. In the second part, Section 2.2, of chapter 2, we provide the background on Neural Networks (NNs), more specifically on CNNs, employed for the classification of audio samples. In chapter 3, we describe the establishment of our proposed dataset based on the Android application, and the framework for the evaluation of the acquired audio samples. In chapter 4, the results, along with an interpretation, for ASC and ALC with local and temporal constraints, employed on the TAU development dataset, and on our proposed dataset, are presented. Chapter 5 concludes this thesis, recapitulating the objective of our work, the approached data collection and evaluation process, and the outcomes of our evaluation. Finally, possible future work is discussed in Section 5.1.

2. Related Work

In this chapter, we provide the required knowledge base for our work. Thus, in the first section, Section 2.1, we introduce the state-of-the-art in the field of artificial audio classification. Furthermore, we provide an in-depth introduction to CNNs in the second section, Section 2.2. Thus, we start off with Feedforward Neural Networks (FFNNs) and transition over to CNNs. Additionally, we introduce concepts utilized along with CNNs, i.e., regularization with dropout and ES, acceleration of the training based on Batch Normalization (BN), and discuss how to achieve reliable results based on Cross-validation (CV). Finally, we provide insights on the optimizer Adaptive Moments (Adam), utilized for our CNN.

2.1. State of the Art

We start off with the dataset utilized for the DCASE 2018 ASC challenge, proposed in [Mesaros2018], representing the predecessor of the dataset employed as benchmark during the evaluation of this thesis.
The dataset consists of audio samples recorded in 6 European cities over a time frame of 2 months. For each city, audio recordings of multiple locations, representing a total of 10 different acoustic scenes, are collected. More specifically, audio recordings with a duration between 2 and 3 minutes at multiple positions in each location are acquired, totalling over 28 hours of audio material. For the development dataset of the DCASE 2018 ASC challenge, 24 hours of stereophonic, more specifically, binaural audio recordings are collected, utilizing an in-ear microphone along with a professional audio recorder, providing a sample rate of 48 kHz and a bit depth of 24 bit. Our work is based on the updated version of this development dataset, proposed for the DCASE 2019 ASC challenge, represented by 40 hours of audio data recorded in 10 European cities.
For the evaluation of the development dataset, a constrained train-test is proposed

in [Mesaros2018]. Hence, the dataset is split into a training and a test set, restricting samples associated with a particular location to either the training or the test set. Furthermore, representatives of each city are distributed throughout the training and the test set. Thus, during the evaluation of this thesis, we are encouraged to investigate the impact of local constraints on the train-test split.
Next, we introduce the baseline CNN, proposed in [Mesaros2018], provided for the DCASE 2018 ASC challenge, and employed during the evaluation of our work. Based on the submission of [Valenti2016] for the DCASE 2016 ASC challenge, [Mesaros2018] implements a 2-layer deep CNN with BN. The features for the network are represented by 40-band log-mel spectrograms of 40 ms windows with an overlap of 20 ms computed on 10 s audio samples.
Opposed to [Mesaros2018], [Valenti2016] utilizes a different number of hidden neurons and 60-band log-mel spectrograms computed on 30 s samples, compared to 40-band log-mel spectrograms computed on 10 s samples. Additionally, [Valenti2016] trains the network on sub-sequences of the log-mel spectrograms, and proposes a learning approach based on 2 stages. In the first stage, denoted as *non-full training*, the training fold is split into a training and a validation set. Then, for each epoch independently, the log-mel spectrograms, computed on 30 s audio samples, are alternately split into non-overlapping sub-sequences, e.g., 3 s long sub-sequences, and the network is trained until convergence, i.e., 100 epochs without improvements in the training and validation accuracy. In the second stage, denoted as *full training*, the network is trained on the whole training fold for the number of epochs which corresponds to the training time during the first stage. Thus, compared to the DCASE 2016 baseline, proposed in [Mesaros2016], which implements a Gaussian Mixture Model (GMM) based on Mel-frequency Cepstral Coefficients (MFCCs), [Valenti2016] achieves an improvement of 8.8% on the proposed 4-fold CV w.r.t. the development dataset proposed for the DCASE 2016 ASC challenge.

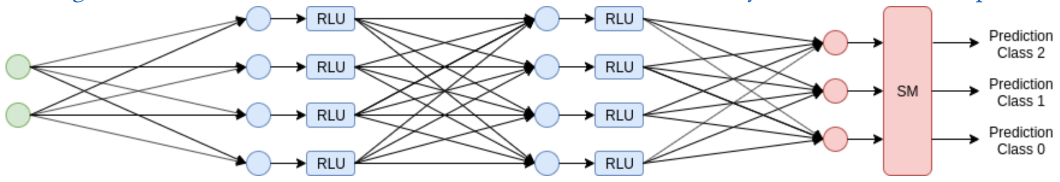
2.2. Background

In this section, we provide the fundamentals of CNNs, employed for ASC and ALC. Thus, we start off with a brief recap of FFNNs in Section 2.2.1, and proceed with an introduction to CNNs in Section 2.2.2. Furthermore, we present regularization based on dropout in Section 2.2.3, followed by BN in Section 2.2.4, and CV in Section 2.2.5. Additionally, we present the application of ES in Section 2.2.6. Finally, the emergence of the employed optimizer, i.e., Adam, is explained in Section 2.2.7.

2.2.1. Feedforward Neural Networks

We start off with a short recap of FFNNs, also known as Multilayer Perceptrons (MLPs), which represent the basis for CNNs. The content is based on the lecture notes of [Legenstein2017] and [Knoebelreiter2017], and Chapter 6, Deep Feedforward Networks, of [Goodfellow2016].

Figure 2.1.: Feedforward Neural Network with 2 hidden ReLU layers and Softmax output.



As illustrated by Figure 2.1, a FFNN consists of an input layer, at least one hidden layer, and an output layer.
In the field of Machine Learning (ML), FFNNs are employed as approximators for non-linear functions. For example, FFNNs with 2 hidden layers, and sufficient hidden neurons, are able to represent any non-linear function. [Legenstein2017]
In general, a FFNN is denoted by a function, $f'(x;\theta) = y'$, which aims to approximate another function $f(x) = y$, where x represents the input, θ the parameters of a network, and y the corresponding output. [Goodfellow2016]
Thus, when training a FFNN, we try to optimize the network parameters θ.
As represented in Figure 2.1, the input of a neuron is composed of all outputs of the previous layer. Furthermore, to introduce non-linearity, hidden activation functions, e.g., Rectified Linear Unit (ReLU), denoted by $h(z) = max\{0, z\}$, are element-wise applied to activate the output of the hidden layers. [Legenstein2017, Goodfellow2016]
Thus, we denote the activated output of a hidden layer by $f^{(i)}(x) = a^{(i)} = h(z^{(i)}) = h(W^{(i)} f^{(i-1)}(x) + b^{(i)})$, where h represents the activation function, W the weights,

b the bias, and i the depth of the particular layer. [Knoebelreiter2017, Goodfellow2016]
Finally, the output of the network depends on the problem statement, hence, on the output activation function. For multi-class classification, which is conducted when classifying audio samples in multiple classes, Softmax, denoted by $h(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$, depicts a common output activation function. [Legenstein2017, Goodfellow2016]
When training a FFNN, the input is forward propagated by computing the activated outputs through the network. The output of the network is then evaluated on a loss function, which provides the error of the output as a result. With Softmax as output activation function, we utilize the categorical cross-entropy loss to evaluate the error, which is denoted by $\mathcal{L}(y, y') = -\sum_i \sum_j y_{ij} ln(y'_{ij})$[1], where y represents the targets, and y' represents the predicted output. The error is then back-propagated through the network by computing the gradient of the loss function w.r.t. the network parameters θ, utilizing the chain rule of calculus. This approach is denoted as Backpropagation, proposed in [Werbos1974, Rumelhart1986]. Finally, the network parameters θ are updated. For example, when using Batch Gradient Descent, the weights are updated by moving in the direction of the negative gradient w.r.t. the corresponding parameter, hence, $\theta = \theta - \eta \nabla_\theta \mathcal{L}(\theta)$, where η represents the learning rate. [Legenstein2017]

2.2.2. Convolutional Neural Networks

Next, we provide an introduction to CNNs, proposed in [Fukushima1980], and [LeCun1989], which we utilize for the classification of audio samples.
CNNs are superior in detecting objects in grid-structured data, e.g., in image classification and time series tasks, and are employed by multiple state-of-the-art solutions. [Goodfellow2016]
Basically, CNNs correspond to FFNNs, but implement an additional type of layer, denoted as convolutional layer.[2]
Conventionally, a pooling layer is placed after the activation of the convolutional

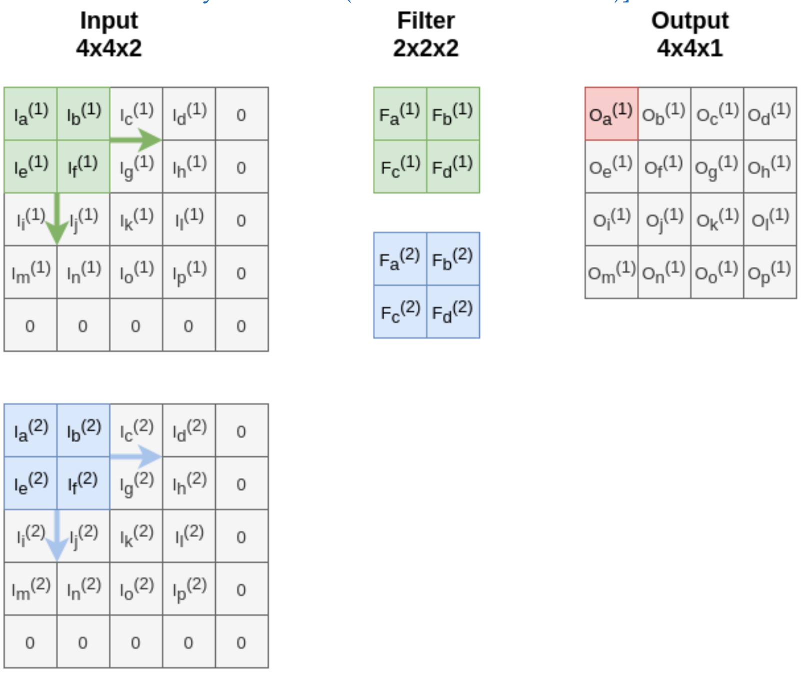
[1] https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy (Accessed on: 2019-09-27)
[2] https://www.youtube.com/watch?v=YRhxdVk_sIs (Accessed on: 2019-09-17)

layer. Thus, there exists another topology, denoting the combination of convolution, activation, and pooling, as convolutional layer. [Goodfellow2016]
However, throughout this thesis, we denote the implementation of convolution, without activation and pooling, as convolutional layer.
Besides, recent results, proposed in [Springenberg2014], suggest to completely replace pooling layers by convolutional layers with a larger stride.
Opposed to FFNNs, CNNs are not fully connected, and the weights, denoted as filters or kernels, are shared when convolving, i.e., sliding, over the input. Thus, the number of network parameters is decreased, and the efficiency, in terms of runtime and memory requirements, is improved. Additionally, CNNs are robust to position changes when applied to detect patterns and objects within the input. On the downside, disparities in the scaling and the rotation of patterns and objects are not handled by default. [Goodfellow2016]
Each convolutional layer consists of one or multiple filters, employed to detect patterns and objects in the input provided to the layer. Starting with the detection of geometric patterns, e.g., edges, in higher layers, and proceeding to more specific objects, e.g., detecting numbers, in deeper layers.[3]
To compute the output of a convolutional layer, also denoted as filter maps, each filter is convolved over the input. Thus, the dot product, i.e., the element-wise product [Karpathy2019], of the filter and the corresponding area of the input, denoted as receptive field, is computed.[3]
Finally, we illustrate the application of convolution in Figure 2.2, utilizing zero-padding to preserve the input dimension for the output, and hence to prevent the loss of information.[4]
Next, we briefly introduce the pooling layer, more precisely, max pooling, which is more relevant in practice compared to average pooling. [Karpathy2019]
In general, pooling computes a summary over the neighborhood of the corresponding input field. Thus, in the case of max pooling, the neighborhood is represented by its maximum value. [Goodfellow2016]
The motivation behind pooling is the reduction of the input dimension for the next layer, resulting in a smaller number of network parameters, and hence in an improved efficiency. [Karpathy2019]
Furthermore, due to the generalized representation of the information, introducing

[3] `https://www.youtube.com/watch?v=YRhxdVk_sIs` (Accessed on: 2019-09-17)
[4] `https://www.youtube.com/watch?v=qSTv_m-KFk0&feature=youtu.be` (Accessed on: 2019-10-23)

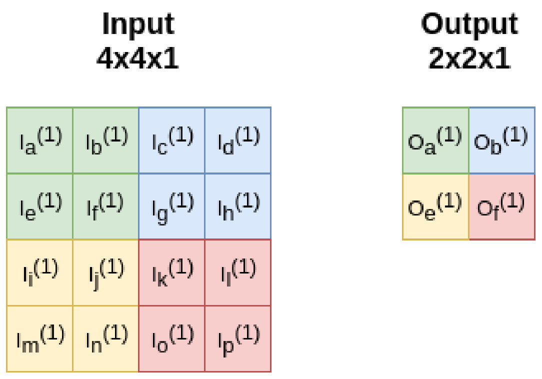
a pooling layer enhances the robustness of the network in respect of overfitting.[5] Thus, in Figure 2.3, we illustrate the application of max pooling.

Figure 2.2.: In this figure, we illustrate the application of convolution on a 3-dimensional input with a single 2×2 filter, and a stride of 1, represented by the horizontal and vertical arrow. Thus, the output is computed by the dot product of the receptive field of the input and the filter. More precisely, the output at position $O_a^{(1)}$ represents the sum of 1) the dot product of the receptive field of the first input layer (green), and the corresponding filter (green), and 2) the dot product of the receptive field of the second layer (blue) with the corresponding filter (blue). The remaining output fields are computed in the same way, while sliding one step at a time over the input.[Goodfellow2016, Karpathy2019] Additionally, to preserve the input dimension for the output, we introduce a zero-padding to the input. [https://www.youtube.com/watch?v=qSTv_m-KFk0&feature=youtu.be (Accessed on: 2019-10-23)]


Input
4x4x2
Filter
2x2x2
Output
4x4x1
Ia(1) Ib(1) Ic(1) Id(1) 0
Ie(1) If(1) Ig(1) Ih(1) 0
Ii(1) Ij(1) Ik(1) Il(1) 0
Im(1) In(1) Io(1) Ip(1) 0
0 0 0 0 0
Ia(2) Ib(2) Ic(2) Id(2) 0
Ie(2) If(2) Ig(2) Ih(2) 0
Ii(2) Ij(2) Ik(2) Il(2) 0
Im(2) In(2) Io(2) Ip(2) 0
0 0 0 0 0
Fa(1) Fb(1)
Fc(1) Fd(1)
Fa(2) Fb(2)
Fc(2) Fd(2)
Oa(1) Ob(1) Oc(1) Od(1)
Oe(1) Of(1) Og(1) Oh(1)
Oi(1) Oj(1) Ok(1) Ol(1)
Om(1) On(1) Oo(1) Op(1)

[5]https://deepai.org/machine-learning-glossary-and-terms/max-pooling (Accessed on: 2019-10-23)

Figure 2.3.: This figure illustrates the application of max pooling, using a pool size of 2×2 and a corresponding slide of 2. E.g., the output $O_a^{(1)}$ presents the maximum value of the corresponding area of the input, i.e., $\max(I_a^{(1)}, I_b^{(1)}, I_e^{(1)}, I_f^{(1)})$.[Karpathy2019]


Input
4x4x1
Output
2x2x1
$I_a^{(1)}$ $I_b^{(1)}$ $I_c^{(1)}$ $I_d^{(1)}$
$I_e^{(1)}$ $I_f^{(1)}$ $I_g^{(1)}$ $I_h^{(1)}$
$I_i^{(1)}$ $I_j^{(1)}$ $I_k^{(1)}$ $I_l^{(1)}$
$I_m^{(1)}$ $I_n^{(1)}$ $I_o^{(1)}$ $I_p^{(1)}$
$O_a^{(1)}$ $O_b^{(1)}$
$O_e^{(1)}$ $O_f^{(1)}$

2.2.3. Dropout Regularization

Dropout represents a regularization methodology for deep NNs to avoid overfitting, hence, is utilized to improve generalization. As proposed in [Srivastava2014], NNs with dropout achieved significant results in various domains, e.g., speech recognition, image, and document classification.
When using dropout, randomly selected neurons are dropped, i.e., are ignored, during the training. More specifically, the input to the neuron, the neuron itself, and the neuron's output are omitted. Therefore, just a particular part of the NN, i.e., a sub-network, is trained each time. On an abstract point of view, a NN utilizing dropout can be compared to multiple smaller sub-networks, each contributing to the prediction. Thus, dropout can be seen as a kind of model averaging. [Legenstein2017, Srivastava2014]

2.2.4. Acceleration with Batch Normalization

BN, proposed in [Ioffe2015], aims to accelerate the training of deep NNs. This acceleration is achieved by reducing the Internal Covariate Shift (ICS), proposed in [Shimodaira2000]. ICS describes a phenomenon induced when using multiple layers:

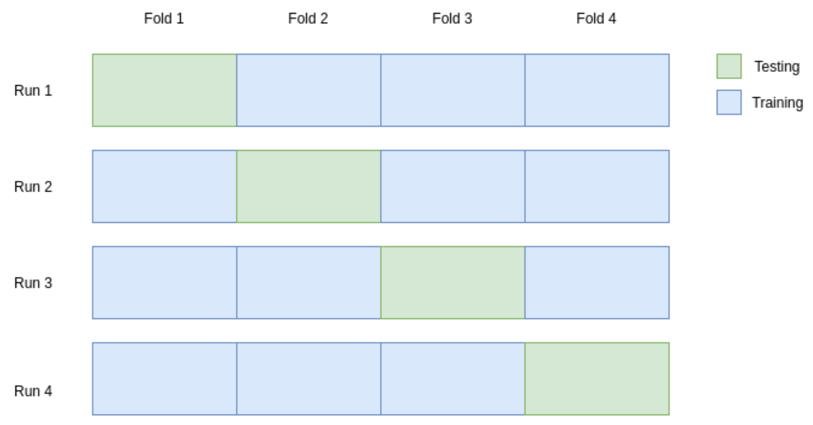
Changes in the parameters of a layer alter the layer's output, thus, the input for the proceeding layer. Consequently, when the distribution of the previous layer changes, the proceeding layer has to adapt to the new distribution. This recursive adaption to new distributions is denoted as ICS.[6]
Especially for very deep networks, this effect results in a slow convergence behavior.
To reduce the ICS, the input to layers is batch-wise normalized. Since this approach allows larger learning rates, the training of deep NNs is accelerated. Furthermore, the impact of the selected parameter initialization methodology becomes less significant. With the utilization of BN, [Ioffe2015] achieved state-of-the-art results on the ImageNet benchmark, proposed in [Deng2009], while keeping the number of training iterations significantly low. [Ioffe2015]

2.2.5. K-Fold Cross-Validation

In the following, we provide a brief introduction to K-fold Cross-validation (K-fold CV), which is utilized to improve the reliability of performance estimates in consideration of ML approaches. As illustrated in Figure 2.4, the dataset is split into multiple parts, more precisely, into k parts, denoted as folds. At each step, $k-1$ folds are utilized for training, and one fold for testing, i.e., for the validation. Thus, the particular model is trained multiple times, but on different sets of samples, and validated against the remaining unseen samples. Ultimately, after k training runs, the arithmetic mean of the test losses, or accuracies, is declared as final result, i.e., as the performance estimate. Especially for datasets with a limited number of samples, this procedure allows to determine a more meaningful estimate of the model's performance. [Legenstein2017]

[6] `https://www.quora.com/Why-does-an-internal-covariate-shift-slow-down-the-training-procedure`, Zhao, Nathan on 2019-05-06 (Accessed on: 2019-10-21)

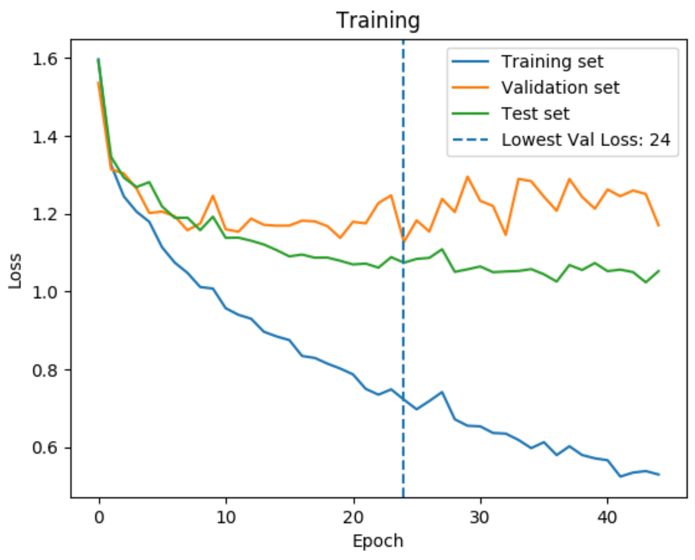
Figure 2.4.: Illustration of a 4-fold CV, representing the train-test splits of the corresponding 4 folds w.r.t. the CV runs.



2.2.6. Early Stopping

Next, we provide a brief introduction to ES, a regularization mechanism to avoid overfitting. When training a NN on a set of samples, i.e., on the training set, the network parameters are adapted w.r.t. the loss function. Thus, the network performance on the training set improves continuously. In practice, we need the network to perform well on unseen samples, too. Unfortunately, when training a network, the performance on unseen samples, i.e., on the test set, tends to become worse after a particular number of epochs. At this moment, we denote the network to start overfitting, i.e., further improving on the training data, while starting to perform worse on unseen samples. Thus, ES is utilized to prevent a NN from overfitting. Therefore, a minor part of the training set is split into a separated validation set. During training, the loss on this validation set is monitored and utilized to determine the best performing model. Thus, we receive a generalized model, performing well on unseen samples. [Legenstein2017, Prechelt1998]
Finally, we present the application of ES in Figure 2.5, illustrating the training, validation, and test loss of employing ASC on the TAU development dataset, while restricting the samples to one of the 3 sets w.r.t. the associated location, as proposed in [Mesaros2018], and stopping the training after 20 epochs of stagnating validation loss.

Figure 2.5.: Training, validation, and test loss of employing ASC with ES on the TAU development dataset. Thus, 70% of the dataset are utilized for training and 30% for testing. Additionally, 10% of the training set is split apart for the validation set. Furthermore, samples are restricted to either the training, validation, or test set w.r.t. to the labelled location. Finally, the training is stopped after 20 epochs of non-improvement w.r.t. the validation loss.



2.2.7. A Journey from Batch Gradient Descent to Adam

In the following, we provide an introduction to the emergence of the optimizer Adam, proposed in [Kingma2014], which we utilize for our CNN. Utilizing Adam is motivated due to the application in the DCASE 2019 baseline CNN and due to the results presented in [Kingma2014]. [Kingma2014] demonstrates the capabilities of Adam by outperforming various state-of-the-art algorithms, i.e., AdaGrad, proposed in [Duchi2011], and RMSProp, proposed in [Tieleman2012, Graves2013], on image classification and sentiment analysis benchmarks.
The content of this introduction is based on the lecture notes of [Legenstein2017], and Chapter 8, Optimization for Training Deep Models, of [Goodfellow2016].
Hence, the remainder of this section is structured as follows. We start off with Batch Gradient Descent (BGD), proceed with Mini-batch Gradient Descent (MBGD), and,

finally, we introduce the improvements induced by Adam.
Since the direction of the negative gradient of the loss function, w.r.t. to the corresponding parameter, represents the direction where the error decreases the most, each of the following optimization algorithms, i.e., BGD, MBGD, and Adam, utilizes the gradient of the loss function to optimize the parameters. [Legenstein2017]
When using BGD, also denoted as Deterministic Gradient Descent (DGD), the gradient of the loss function is computed over all training samples. [Legenstein2017, Goodfellow2016]
Therefore, in Algorithm 1, we present the optimization algorithm BGD.

Algorithm 1 Batch Gradient Descent [Legenstein2017]

```
1: $\theta \leftarrow$ init()                          ▷ initialize parameters
2: while stopping criterion not reached do
3:     $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta)$     ▷ compute gradient on all samples
```

Next, we present MBGD, also denoted as Mini-batch Stochastic Gradient Descent (MBSGD), and Stochastic Gradient Descent (SGD). However, historically, SGD is identified by gradient updates based on a single sample. [Goodfellow2016]
MBGD does not compute the gradient over all training samples, but updates the parameters on small batches of samples, e.g., on 32, 16, or 8 samples. MBGD allows the computation of batch updates in parallel, thus, can accelerate the training runtime. Additionally, MBGD converges faster and is able to escape local minima. [Legenstein2017]
According to [Masters2018], the batch size, i.e., the number of samples per batch, should not exceed 32 samples. Emphasized by *Yann LeCun*, who introduced todays CNNs in [LeCun1998], with a post on the social media platform Twitter: *"Training with large minibatches is bad for your health. More importantly, it's bad for your test error. Friends dont let friends use minibatches larger than 32."*[7]
Furthermore, keeping the batch size a magnitude of 2 might improve the performance of particular implementations.[8]

[7] https://twitter.com/ylecun/status/989610208497360896?lang=en (Accessed on: 2019-10-01)
[8] https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3 (Accessed on: 2019-11-04)

Thus, MBGD is presented in Algorithm 2.

Algorithm 2 Mini-Batch Gradient Descent [Goodfellow2016]

```
1: $\theta \leftarrow$ *init()*                                                        ▷ initialize parameters
2: while *stopping criterion not reached* do
3:     $\theta \leftarrow \theta - \eta \frac{1}{\mathcal{N}} \sum_{i=0}^{\mathcal{N}} \nabla_\theta \mathcal{L}(y_i; f(x_i; \theta))$    ▷ compute gradient on mini-batch
```

Finally, we present Adam, proposed in [Kingma2014], in Algorithm 3. Adam belongs, among AdaGrad, proposed in [Duchi2011], and RMSProp, proposed in [Tieleman2012, Graves2013], to the category of optimization algorithms which implement adaptive learning rates. Thus, AdaGrad scales the learning rate with the accumulated inverted roots of the squared gradients. RMSProp corresponds to AdaGrad, but implements a running average for the accumulation. Additionally to an adaptive learning rate, Adam is designed to accelerate learning based on momentum, proposed in [Polyak1964]. [Goodfellow2016]
Thus, in general, Adam can be compared to RMSProp with momentum, according to [Kingma2014].
But, opposed to RMSProp, Adam implements momentum directly into the estimate of the first *moment*. Additionally, Adam corrects the bias of the *moment* estimates. [Goodfellow2016]

Algorithm 3 Adaptive Moments (Adam) [Goodfellow2016, Kingma2014]:
The provided initialization values represent the default values proposed in [Kingma2014]. p_1^τ denotes p_1 to the power of τ, and p_2^τ denotes p_2 to the power of τ, respectively.

```
1:  $\tau \leftarrow 0$                                          ▷ time step
2:  $\eta \leftarrow 0.001$                                      ▷ step size
3:  $p_1, p_2 \leftarrow 0.9, 0.999$                             ▷ exponential decay rates $p_1, p_2 \in [0, 1)$
4:  $m_1, m_2 \leftarrow 0, 0$                                   ▷ $1^{st}$ and $2^{nd}$ moment estimate
5:  $\epsilon \leftarrow 10^{-8}$                                ▷ constant for numerical stability
6:  $\theta \leftarrow$ *init()*                                 ▷ initialize parameters
7:  **while** *stopping criterion not reached* **do**
8:      $\tau \leftarrow \tau + 1$                               ▷ increment time step
9:      $g \leftarrow \frac{1}{N} \sum_{i=0}^{N} \nabla_\theta \mathcal{L}(y_i; f(x_i; \theta))$   ▷ compute gradient on mini-batch
10:     $m_1 \leftarrow p_1 m_1 + (1 - p_1) g$                   ▷ compute $1^{st}$ moment estimate
11:     $m_2 \leftarrow p_2 m_2 + (1 - p_2) g \odot g$           ▷ compute $2^{nd}$ moment estimate
12:     $m_1' \leftarrow \frac{m_1}{1 - p_1^\tau}$                ▷ correct bias of $1^{st}$ moment estimate
13:     $m_2' \leftarrow \frac{m_2}{1 - p_2^\tau}$                ▷ correct bias of $2^{nd}$ moment estimate
14:     $\theta \leftarrow \theta - \eta \frac{m_1'}{\sqrt{m_2' + \epsilon}}$   ▷ update parameters
```

3. Methodology

In this chapter, we present our approach for the collection and evaluation of audio recordings for ASC and ALC. Thus, we start off with an overview of the infrastructure, utilized to establish our proposed dataset, in Section 3.1. Therefore, we provide an in-depth introduction to the employed Android application in Section 3.1.1, and present the web service, and the database, in Section 3.1.2, and Section 3.1.3, respectively. In Section 3.2, we introduce the established dataset, along with the TAU development dataset, utilized as benchmark for the evaluation. Finally, in Section 3.3, we present the classification methodology, providing a detailed introduction to the feature extraction, i.e., log-mel spectrograms, in Section 3.3.1, and introduce the employed CNN in Section 3.3.2.

3.1. Infrastructure

In this section, we present the infrastructure, consisting of an Android application, and a Node.js Representational State Transfer (REST)[1]web service, along with a SQLite database, employed to establish a dataset for the investigation of ASC and ALC.
To provide the possibility of collecting and sharing audio recordings to as many individuals as possible, we decide to implement an Android application over an iOS Application. This decision is based on the current market share of smartphone operating systems, i.e., Android excels iOS with 76% to 22%.[2]Unfortunately, web applications are not capable of directly accessing native device resources, and hence are not suited to gather reliable audio recordings. Thus, we do not consider

[1]https://en.wikipedia.org/wiki/Representational_state_transfer (Accessed on: 2019-11-18)
[2]https://gs.statcounter.com/os-market-share/mobile/worldwide (Accessed on: 2019-09-04)

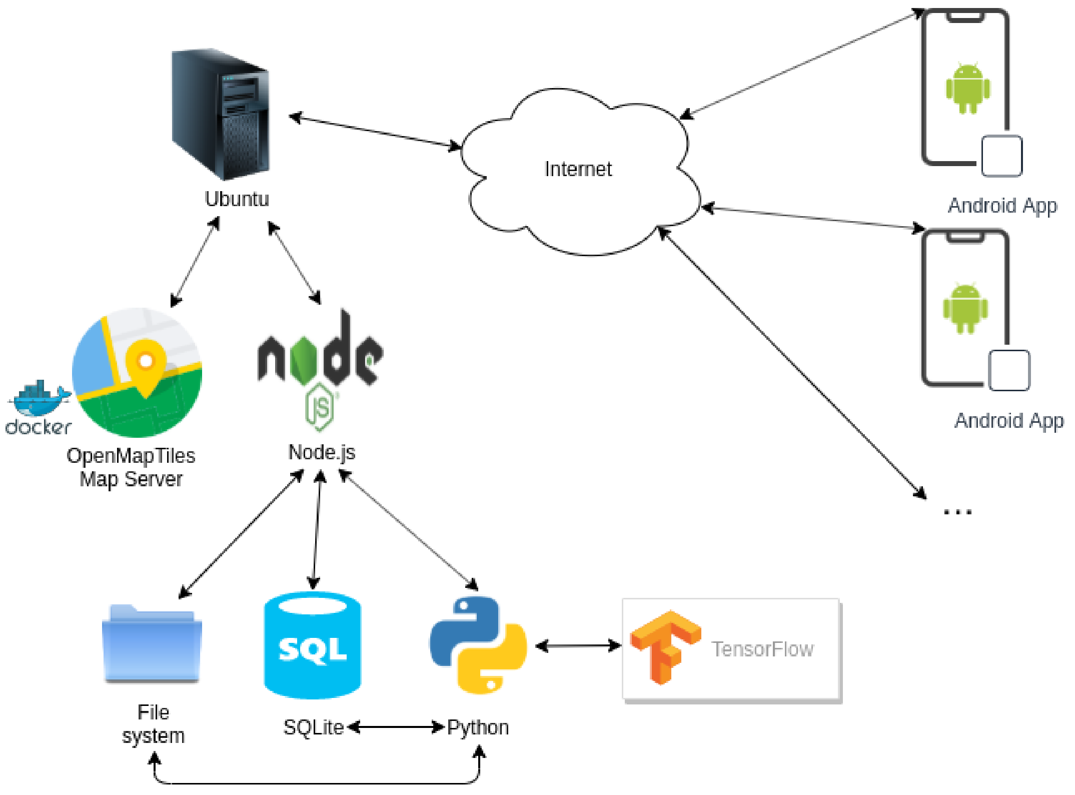
web applications as an appropriate alternative.
We declare a minimum Software Development Kit (SDK) version of 24 (Android 7, Nougat), and a target SDK version of 28 (Android 9, Pie).[3]Hence, with a minimum SDK version of 24 , we support approximately 80% of all Android devices.[4]Furthermore, the support of Android itself is lower bound by the SDK version of 24.[5]At the start of this thesis (Mai, 2019), the target SDK version of 28, i.e., Android Pie, represents the most recent version.
For the development, we utilize a Nexus 6 with Android 7.1.1. Thus, a flawless employment of the application is only guaranteed for the denoted device and Android version.
To avoid restrictions on the distribution of the Android application, we decide to publish the application via a website, instead of the Google Play Store. Thus, the website is hosted on a dedicated server and can be accessed via *stck.ddns.net*. Furthermore, the address resolution is realized with the dynamic Domain Name System (DNS) service provided by *noip.com*.
To publish audio recordings remotely, we utilize a Node.js (v11.15.0) REST web service, hosted on our dedicated server. Furthermore, we provide a SQLite (v3.22.0) database to gather additional metadata, along with the uploaded audio recordings stored on the file system. On top of that, the database is employed for the user management, representing a mandatory requirement due to the current legal situation. Additionally, the web service provides the evaluation of uploaded audio recordings w.r.t. ASC and ALC based on a pre-trained CNN.
Summarized, in Figure 3.1, we illustrate the established infrastructure.

[3] https://developer.android.com/studio/releases/platforms (Accessed on: 2019-09-04)

[4] https://gs.statcounter.com/android-version-market-share/mobile-tablet/worldwide (Accessed on: 2019-09-04)

[5] https://en.wikipedia.org/wiki/Android_version_history (Accessed on: 2019-09-04)

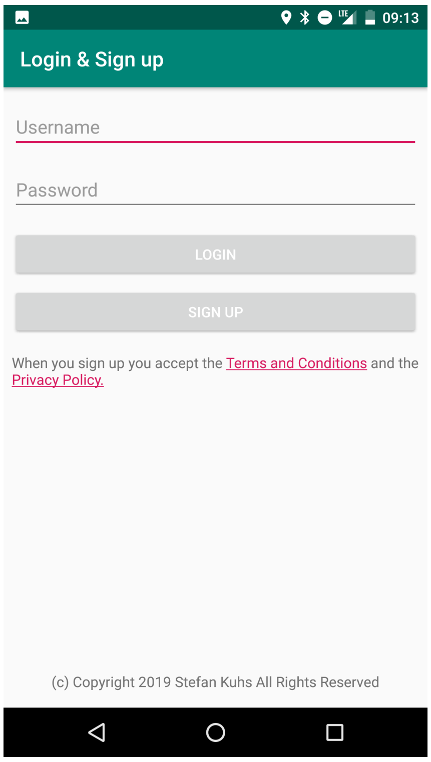
Figure 3.1.: The infrastructure employed to gather audio recordings along with the required metadata, represented by Android smartphones running an instance of our application, and hence communicating with the Node.js REST web service hosted on our server. Thus, the web service provides an interface to store audio recordings on the file system along with metadata in the SQLite database. Furthermore, to approach ASC and ALC on uploaded audio recordings, the web service spawns a Python process and employs a pre-trained CNN on demand. In addition, an OpenMapTiles Map Server, running in a Docker container, provides tiles for the target region, i.e., Austria.



Next, we provide an in-depth introduction to the Android application in Section 3.1.1, and specify the web service, and the database, in Section 3.1.2, and in Section 3.1.3, respectively.

3.1.1. Android Application

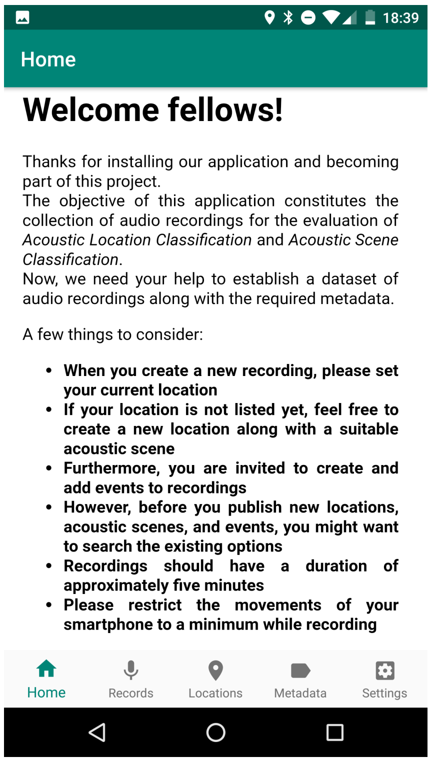
The motivation behind our Android application is driven by the establishment of an audio dataset for the evaluation of ALC and ASC. Hence, to enable the collection of reliable audio data, and to ensure the availability of the required metadata, we provide a dedicated software solution for mobile devices. From there, the participation in the collection process is available to every person with a compatible Android smartphone.
Thus, in the following, we illustrate the functionalities and features offered by our Android application. Additionally, we provide an in-depth introduction to the approached methodology of capturing audio, and discuss the implementation of secondary features, i.e., location tracking, and the visualization of geographic data.



When you sign up you accept the Terms and Conditions and the Privacy Policy.

(c) Copyright 2019 Stefan Kuhs All Rights Reserved

Figure 3.2.: To use our application, and hence to contribute to the collection of audio recordings, a user signs up by providing a unique username and a password. After signing up, we enable users to log in with the initially provided user credentials.



Welcome fellows!

Thanks for installing our application and becoming part of this project.
The objective of this application constitutes the collection of audio recordings for the evaluation of *Acoustic Location Classification* and *Acoustic Scene Classification*.
Now, we need your help to establish a dataset of audio recordings along with the required metadata.

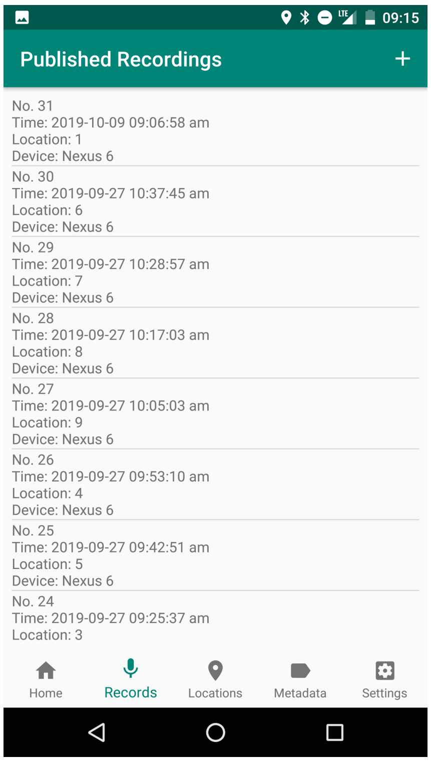
A few things to consider:

- **When you create a new recording, please set your current location**
- **If your location is not listed yet, feel free to create a new location along with a suitable acoustic scene**
- **Furthermore, you are invited to create and add events to recordings**
- **However, before you publish new locations, acoustic scenes, and events, you might want to search the existing options**
- **Recordings should have a duration of approximately five minutes**
- **Please restrict the movements of your smartphone to a minimum while recording**

Figure 3.3.: On the home screen, we provide a brief introduction to our application, thus, we briefly explain the goal of our work, and provide guidelines for contributing to the collection of audio recordings. Additionally, the home screen enables the propagation of recent news throughout the community.

First, we introduce the registration, and the login, respectively, presented in Figure 3.2. To employ our Android application, and hence to participate in the collection of audio recordings, a user signs up by providing a unique username and a password. A mandatory registration enables the association between users and the contributed objects, i.e., recordings, locations, acoustic scenes, and events, and hence provides the framework to administer data in respect of data protection

regulations. Additionally, on registration, the corresponding user agrees to the *Terms and Conditions* and the *Privacy Policy*. After registration, a user logs in with the user credentials, i.e., username and password, provided on registration. To avoid repetitive prompts for user credentials, and hence to improve the user experience, a user remains logged in, until explicitly logged out.
After signing up, and logging in, respectively, the user is forwarded to the home screen, presented in Figure 3.3. Therefore, we briefly introduce the objective of this application, and provide guidelines for the contribution to the collection of audio recordings. Additionally, the home screen facilitates the propagation of recent news, the latest features, and current bugs.



Published Recordings

No. 31
Time: 2019-10-09 09:06:58 am
Location: 1
Device: Nexus 6

No. 30
Time: 2019-09-27 10:37:45 am
Location: 6
Device: Nexus 6

No. 29
Time: 2019-09-27 10:28:57 am
Location: 7
Device: Nexus 6

No. 28
Time: 2019-09-27 10:17:03 am
Location: 8
Device: Nexus 6

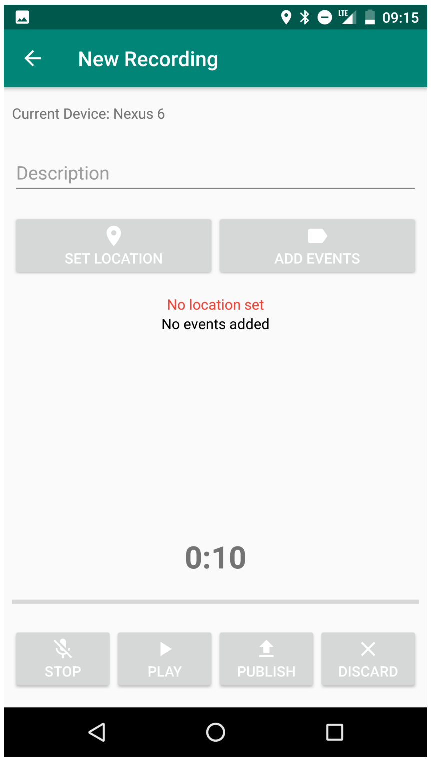
No. 27
Time: 2019-09-27 10:05:03 am
Location: 9
Device: Nexus 6

No. 26
Time: 2019-09-27 09:53:10 am
Location: 4
Device: Nexus 6

No. 25
Time: 2019-09-27 09:42:51 am
Location: 5
Device: Nexus 6

No. 24
Time: 2019-09-27 09:25:37 am
Location: 3

Figure 3.4.: The published audio recordings of a user, providing the object identifier, and the timestamp of conducting the recording, along with the associated location, and the device model identifier of the corresponding smartphone. Optionally, a description, and events captured during the recording, are listed.



New Recording

Current Device: Nexus 6

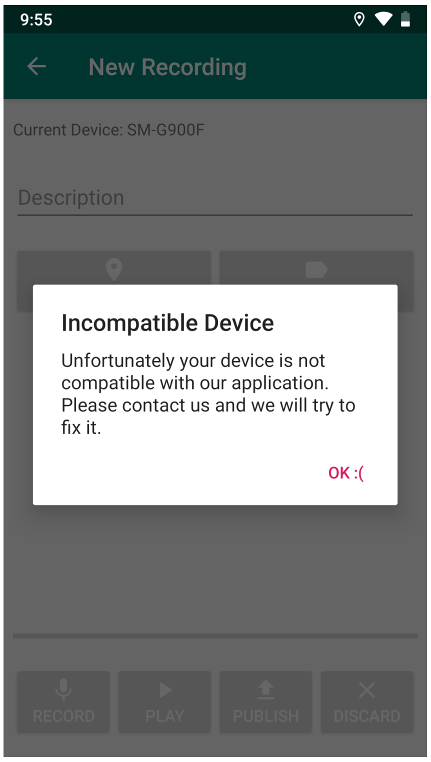
No location set
No events added

0:10

Figure 3.5.: To publish an audio recording, the user provides at least the recorded location. Optionally, a description, and events captured during the recording, might be added. Besides, we enable the user to replay the audio recording before the publication, and to discard the audio recording.

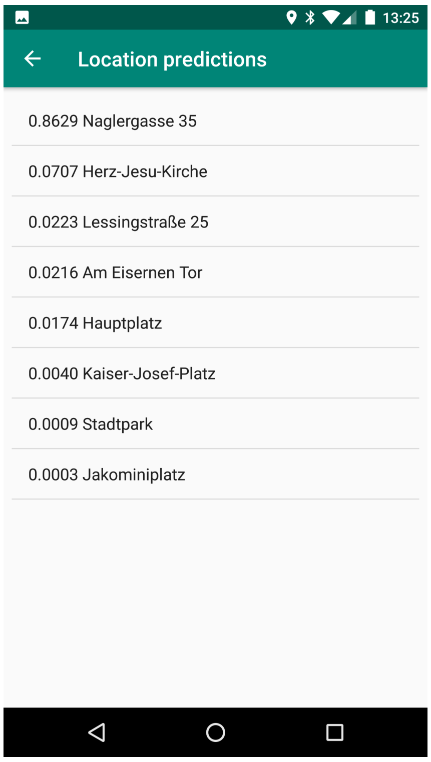
Next, we present the published audio recordings of a particular user in Figure 3.4. Thus, for each audio recording, we display the corresponding object identifier, the timestamp when the recording is conducted, the associated location, and the device model identifier of the smartphone used to capture the audio recording. Optionally, a description, and events associated with the audio recording, are listed. However, due to privacy concerns, the access to published audio recordings is restricted to

the originator only.
To create and publish an audio recording, as presented in Figure 3.5, a user selects the location where the recording is conducted, and might provide events captured during the recording. Hence, the selected location has to correspond to the user's geographic position. Furthermore, a textual description of the audio recording might be provided. Additionally, we enable the user to pause and resume a recording, to replay captured audio, and to discard unpublished data.
In case, the employed Android device does not support the recording of unprocessed audio with the required properties, i.e., Pulse-code Modulation (PCM) encoding with a bit depth of 16 bit, and a sample rate of 48 kHz, recording, and hence publishing of audio is disabled.
For example, the Samsung Galaxy S5 does not provide unprocessed audio, thus, the corresponding functionality is not available, as presented in Figure 3.6.


Incompatible Device

Unfortunately your device is not compatible with our application. Please contact us and we will try to fix it.

Figure 3.6.: Android devices without the ability to record unprocessed audio data with a bit depth of 16 bit, and a sample rate of 48 kHz are not fully supported by our application. Thus, recording, and hence publishing of audio is not available on, e.g., the Samsung Galaxy S5.



13:25

Location predictions

0.8629 Naglergasse 35

0.0707 Herz-Jesu-Kirche

0.0223 Lessingstraße 25

0.0216 Am Eisernen Tor

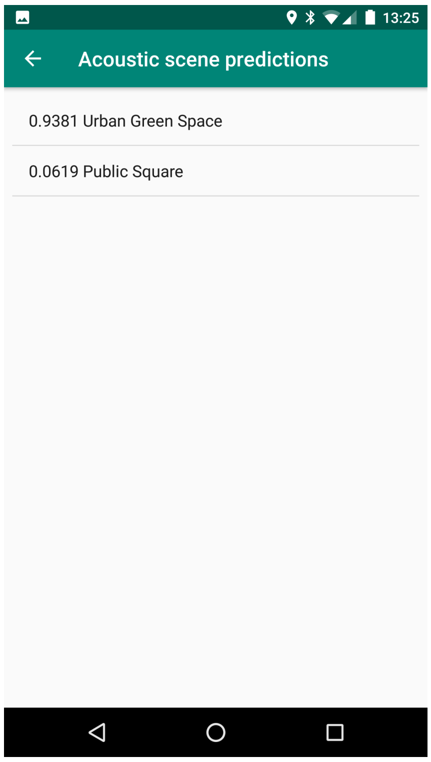
0.0174 Hauptplatz

0.0040 Kaiser-Josef-Platz

0.0009 Stadtpark

0.0003 Jakominiplatz

Figure 3.7.: To supply users with feedback on uploaded audio recordings, we enable ALC with a CNN trained on our proposed dataset, and hence provide the classification predictions of audio samples w.r.t. the recorded location.



13:25

Acoustic scene predictions

0.9381 Urban Green Space

0.0619 Public Square

Figure 3.8.: Additionally to ALC, we enable ASC on uploaded audio recordings. Thus, we provide the classification predictions of audio samples w.r.t. acoustic scenes, i.e., *Urban Green Space* and *Public Square*.

Furthermore, to provide feedback on published audio recordings, we enable users to employ ALC, and ASC, on the corresponding audio samples, i.e., 10 s sequences of the recordings. Therefore, we classify the audio samples with a CNN trained on the samples of our proposed dataset. Thus, this feature is unlocked after the dataset has been established.
More specifically, on selecting a published audio recording, the corresponding file on the server is split into 10 s sequences, the 40-band log-mel spectrograms are extracted, the classification predictions are evaluated based on the particular

CNN model—either classifying the samples to locations, or to acoustic scenes—and, finally, the arithmetic mean of the predictions is provided.
Thus, we present the results of employing ALC, and ASC, on an approximately 300 s audio recording of *Naglergasse 35*, in Figure 3.7, and Figure 3.8, respectively.

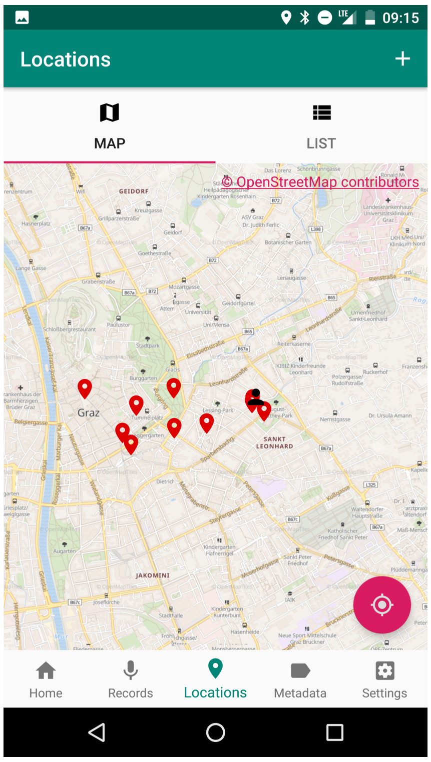
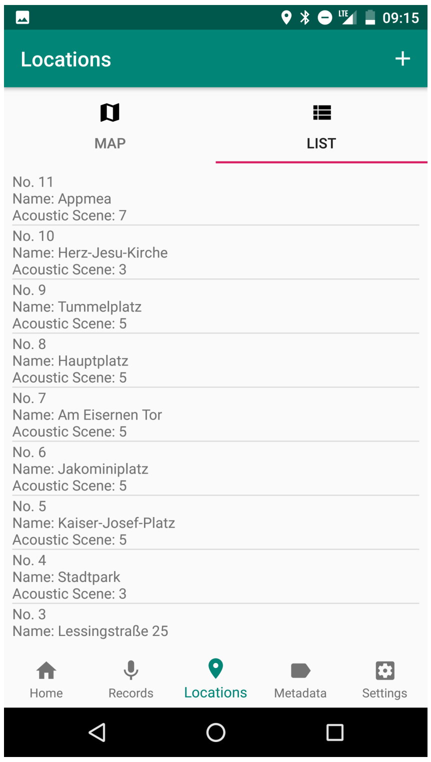


Figure 3.9.: The contributed locations, and the position of the current user, provided on a map view. To receive multiple audio recordings of the same location, the published locations are shared among the users.



No. 11
Name: Appmea
Acoustic Scene: 7

No. 10
Name: Herz-Jesu-Kirche
Acoustic Scene: 3

No. 9
Name: Tummelplatz
Acoustic Scene: 5

No. 8
Name: Hauptplatz
Acoustic Scene: 5

No. 7
Name: Am Eisernen Tor
Acoustic Scene: 5

No. 6
Name: Jakominiplatz
Acoustic Scene: 5

No. 5
Name: Kaiser-Josef-Platz
Acoustic Scene: 5

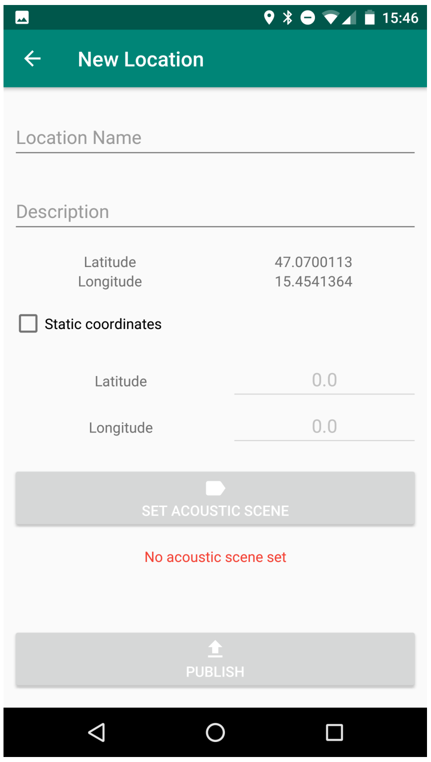
No. 4
Name: Stadtpark
Acoustic Scene: 3

No. 3
Name: Lessingstraße 25

Figure 3.10.: The published locations, along with the corresponding acoustic scenes, optionally, with an additional description, provided by, and shared throughout all users.

Next, we present the published locations, contributed to motivate the publishing of locally concentrated audio recordings. Hence, to receive multiple recordings of the same location, we share the locations between the participants. Therefore, we provide a map view of the locations, along with the current geographic position of the particular user, as presented in Figure 3.9. Additional functionality of the map view implements zooming in, and out, centering the map, and, additionally, locking the center of the map to the user's geographic position. Furthermore, the location identifier, the name of the location specified by the user, and the associated acoustic scene are provided within the corresponding list entry, as presented in Figure 3.10. Optionally, a textual description of the location is available.

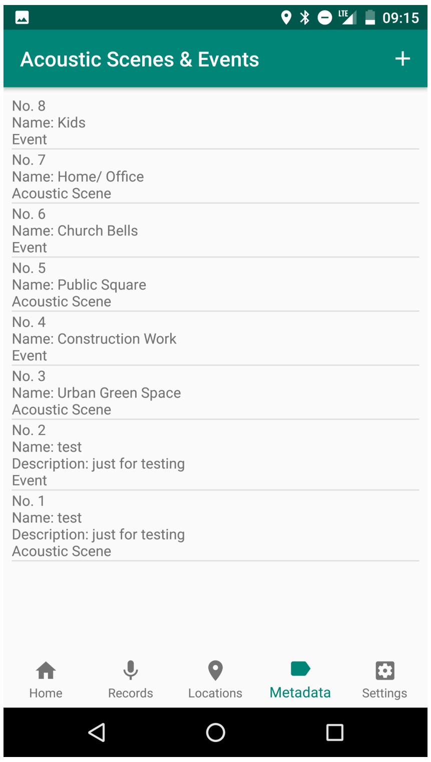
To publish a location, as presented in Figure 3.11, the user provides the name of the corresponding location, and, optionally, a textual description. Additionally, the acoustic scene, associated with the location, is specified by the user. To enable high accuracy position specifications, especially for indoor locations, we allow users to manually provide the geographic position if necessary.



Latitude	47.0700113
Longitude	15.4541364

Latitude	0.0
Longitude	0.0

Figure 3.11.: To publish a location, the user provides the corresponding name, and might offer an additional description. Furthermore, exactly one acoustic scene is selected for the location. Additionally, to provide more precise location specifications, e.g., for indoor locations, users might manually provide the geographic position.


Acoustic Scenes & Events

No. 8
Name: Kids
Event

No. 7
Name: Home/ Office
Acoustic Scene

No. 6
Name: Church Bells
Event

No. 5
Name: Public Square
Acoustic Scene

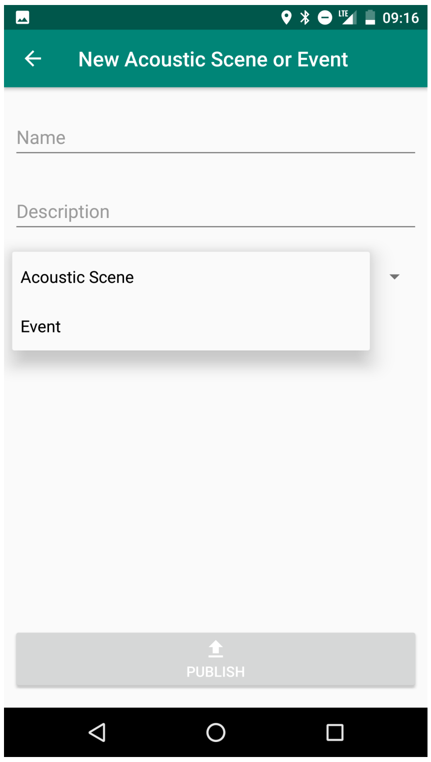
No. 4
Name: Construction Work
Event

No. 3
Name: Urban Green Space
Acoustic Scene

No. 2
Name: test
Description: just for testing
Event

No. 1
Name: test
Description: just for testing
Acoustic Scene

Figure 3.12.: The published acoustic scenes, and events, representing the available metadata for locations, and audio recordings, respectively. To receive comparable audio recordings, w.r.t. the provided metadata, we share the published acoustic scenes, and events, throughout all users.


New Acoustic Scene or Event

Name

Description

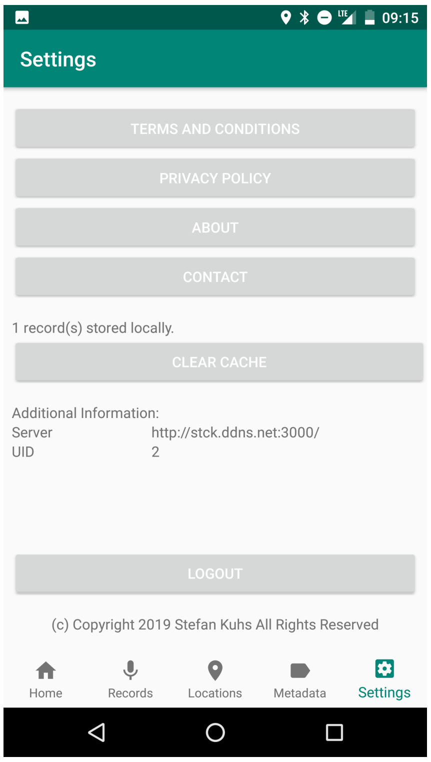
Acoustic Scene

Event

Figure 3.13.: To publish acoustic scenes, and events, respectively, the particular user provides an appropriate name, and, optionally, a more detailed description to clarify the objective.

Additionally, we present the published acoustic scenes and events, contributed to provide standardized metadata for publishing locations, and audio recordings, respectively. Thus, along with locations, we share acoustic scenes and events between the participants, as presented in Figure 3.12.
To publish an acoustic scene or event, the user provides a meaningful name, optionally, a textual description, and specifies the category, i.e., acoustic scene or

event, as presented in Figure 3.13.



1 record(s) stored locally.

Additional Information:

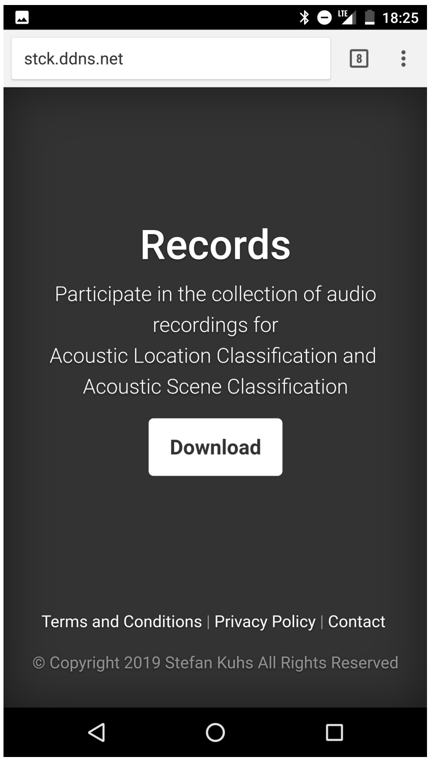
Server	http://stck.ddns.net:3000/
UID	2

(c) Copyright 2019 Stefan Kuhs All Rights Reserved

Figure 3.14.: In the settings, we provide links to legal documents, i.e., *Terms and Conditions*, and the *Privacy Policy*, and to our website, along with contact details. Additionally, we allow the deletion of locally cached audio recordings, and provide the possibility to log the current user out.

Next, we present the settings of our application in Figure 3.14.
Thus, we grant access to legal documents, i.e., *Terms and Conditions*, and the *Privacy Policy*, refer to the project's website, and provide our contact details.
While recording, the audio data is directly streamed to the internal storage of the device, and preserved until explicitly removed. Hence, due to the limited internal storage of smartphones, locally cached audio recordings can be manually deleted.
Furthermore, we allow users to log out, and hence enable multiple contributors on the same device.

Finally, we introduce the website for the public distribution of our Android application. Thus, in Figure 3.15, we present the homepage, providing the download for the Android application, links to legal documents, i.e., *Terms and Conditions*, and the *Privacy Policy*, along with contact details.



Records

Participate in the collection of audio recordings for
Acoustic Location Classification and Acoustic Scene Classification

Terms and Conditions | Privacy Policy | Contact

© Copyright 2019 Stefan Kuhs All Rights Reserved

Figure 3.15.: The homepage provided for the public distribution of our Android application. Additionally, we grant access to legal documents, i.e., *Terms and Conditions*, and the *Privacy Policy*, along with contact details.

Recording and Playing Audio on Android

In the following, we describe the methodology of recording audio on our Android application, and hence discuss crucial design decisions concerning the implementation.

To record, and play audio data on Android[6], we implement *AudioRecord*[7], and *AudioTrack*[7], respectively. Additionally, to preserve raw audio data, we employ the audio source *UNPROCESSED*[8]. Furthermore, we verify the support for unprocessed audio recordings on the corresponding property[9], i.e., *PROPERTY_SUPPORT_-AUDIO_SOURCE_UNPROCESSED*. Otherwise, the audio source *DEFAULT* might be unintentionally employed.
Thus, we ensure, at least to a certain degree, consistent properties for recording audio on different devices, and avoid undesired preprocessing. Therefore, our application is limited to devices with the ability to record unprocessed audio.
In general, Android supports the recording of stereophonic audio data. However, it might depend on the particular device, and the manufacturer, whether, and if so, to which extend the recording of stereophonic audio is available, and reliable.[10][11]
Thus, we restrict our application to the recording of monophonic audio.
Additionally, since Android devices do not necessarily support the streaming, i.e., playing, of monophonic audio signals, e.g., the Nexus 6, we artificially assemble stereophonic output by providing the recorded monophonic audio signal on both channels.
Furthermore, we feature the maximum sample rate supported by the Nexus 6[12], i.e., 48 kHz, and provide the maximum bit depth for integer PCM encoding supported by Android 9, i.e., 16 bit.
To store the recorded audio, we directly stream the PCM data to the internal storage of the device, thus, the application is not restricted by the available Random Access Memory (RAM). Furthermore, once an audio recording is completed, the corresponding PCM file might be uploaded to the server.

[6]https://newventuresoftware.com/blog/record-play-and-visualize-raw-audio-data-in-android.html (Accessed on: 2019-11-24)
[7]https://developer.android.com/reference/android/media/AudioRecord (Accessed on: 2019-11-24)
[8]https://developer.android.com/reference/android/media/MediaRecorder.AudioSource.html#UNPROCESSED (Accessed on: 2019-10-24)
[9]https://developer.android.com/guide/topics/media/mediarecorder#audiocapture (Accessed on: 2019-11-24)
[10]https://stackoverflow.com/questions/21810773, Michael on 2014-02-16 (Accessed on: 2019-11-24)
[11]https://stackoverflow.com/q/14131468, Pfitzinger, Hartmut on 2014-01-27 (Accessed on: 2019-11-24)
[12]https://stackoverflow.com/a/32400479/6833106 (Accessed on: 2019-09-13)

Since we store, and upload PCM files, the required storage and the network traffic are reduced by the Waveform Audio File Format (WAV) headers, and hence 44 bytes are economized per audio recording .[13]

Location Tracking and Maps on Android

Next, we introduce location tracking, along with the visualization of map tiles and locations, on Android.
To track the user's location, we implement the Google Play service Application Programming Interface (API), i.e., the *Fused Location Provider Client*[14]. On a lower level, the Android API implements the *Location Manager*[15], maintaining location providers based on Global Positioning System (GPS) as well as on surrounding networks. However, Google's implementation automatically instructs the appropriate location provider to determine the current location. Furthermore, the location tracking might be prioritized based on the underlying requirements, i.e., accuracy or battery life.[16]
Thus, for our application, we emphasize accuracy over battery life.
To announce the last known location throughout the Android application, a singleton, acting as observable, is implemented. Thus, *Activities*[17]depending on the user's location act as observer, and hence receive location updates once available.
Since *osmdroid*[18]is open source and entirely free of charge, compared to Google's MapView, we utilize the open source project for the visualization of map tiles and locations on our Android application.
Furthermore, to improve the availability of tiles, we host a dedicated *OpenMapTiles* server, providing *OpenStreetMap* tiles for Austria. Since our Android application is

[13]`http://soundfile.sapp.org/doc/WaveFormat` (Accessed on: 2019-09-13)
[14]`https://developers.google.com/android/reference/com/google/android/gms/location/FusedLocationProviderClient` (Accessed on: 2019-11-18)
[15]`https://developer.android.com/reference/kotlin/android/location/LocationManager` (Accessed on: 2019-11-18)
[16]`https://stackoverflow.com/questions/33022662/android-locationmanager-vs-google-play-services` (Accessed on: 2019-11-18)
[17]`https://developer.android.com/reference/android/app/Activity` (Accessed on: 2020-02-29)
[18]`https://github.com/osmdroid/osmdroid` (Accessed on: 2019-11-18)

developed for educational purposes, the *OpenStreetMap* data is provided free of charge by *OpenMapTiles.org*[19].

3.1.2. Web Service

Next, we briefly introduce the Node.js web service, implemented to provide an interface for the Android application to access the database, enable the user management, publish audio recordings, and remotely employ ALC, along with ASC, on uploaded audio recordings.
Thus, to provide the user management, we implement the registration and login of users. On registration, the user provides a unique username and a password. Consequently, on the web service, a pseudo random token is generated, and returned, along with the user id, to grant access to subsequent requests. On login, the credentials, i.e., the username and the password, are verified, and the token, generated on registration, and stored in the database, is returned to the user, along with the corresponding user id. However, the token does not provide any additional security, but provides the framework for more sophisticated methodologies, e.g., JSON Web Tokens (JWTs). To establish a dataset of audio recordings, along with the necessary metadata, we support the creation of locations, acoustic scenes, events, and recordings. Thus, we introduce the corresponding entities, along with their relationships, subsequently, in Section 3.1.3.
However, the main task of the web service constitutes the creation, and upload, of audio recordings. Thus, to publish an audio recording, we first create the corresponding database entry. On successful creation, the audio recording, i.e., the PCM file, is uploaded to the file system of the server.[20]Finally, after the PCM file has been successfully uploaded, the corresponding database entry is validated. Thus, we prevent incomplete audio recordings within the dataset.
Finally, as a special feature, we implement the functionality to employ ALC, along with ASC, on uploaded audio recordings, based on a pre-trained CNN.
However, since the priority of this work represents the establishment of a dataset, and the evaluation of ALC, along with ASC, to investigate temporal and local constraints on training and test sets, the functionality provided by the web service is

[19]https://openmaptiles.org/ (Accessed on: 2019-11-18)
[20]https://stackoverflow.com/questions/9630430, Selva, Andro on 2012-03-12 (Accessed on: 2019-11-25)

limited to the basic requirements. Thus, the implementation of additional features, especially, w.r.t. security, i.e., state-of-the-art security concepts, and authorization, is denoted as future work.

3.1.3. Database

Finally, we provide an overview of the SQLite database. Thus, the database schema is illustrated by the Entity Relationship Diagram (ERD) in Figure 3.16.
The actual implementation might differ in terms and definitions. Furthermore, primary keys are realized as *ROWID*, and foreign keys are not explicitly declared. Additionally, *Acoustic Scenes*, and *Events*, are generalized to one entity, i.e., *Metadata*. However, due to the iterative development, and shifting requirements, we denote further modifications to the database schema as future work.

3.2. Datasets

In this section, we present the datasets utilized for the evaluation of ASC and ALC. Thus, we introduce the TAU development dataset, provided for the DCASE 2019 ASC challenge and employed as benchmark for our work, in Section 3.2.1, and proceed with our proposed dataset, especially established to investigate the practicality and limitations of ALC, in Section 3.2.2.

3.2.1. TAU Urban Acoustic Scenes 2019, Development dataset

The TAU development dataset[21], based on [Mesaros2018], contains audio recordings of 10 acoustic scenes gathered in 10 European cities. For each acoustic scene, multiple audio recordings at various positions in different locations are collected. The hardware solution for recording consists of an in-ear microphone along with a professional audio recorder. Thus, the audio recordings feature a bit depth of 24 bit, a sample rate of 48 kHz, and 2 channels. Each audio recording has a duration between 2 and 3 minutes and is recorded with minimal microphone movement.

[21]http://dcase.community/challenge2019/task-acoustic-scene-classification (Accessed on: 2019-09-23)

Due to privacy concerns, audio recordings containing sensitive information are excluded. Finally, each audio recording is split into 10 s sequences. Thus, the dataset consists of 14400 sequences (40 hours), totalling 41.5 GB.
The TAU development dataset is provided for the DCASE 2019 ASC challenge, hence, is intended for ASC related tasks. However, besides ASC, we investigate the TAU development dataset for the application of ALC. Unfortunately, the dataset neither provides audio recordings of the same location on different dates, nor the necessary metadata to determine the corresponding recording for particular sequences. Thus, sequences drawn from the same recording might be distributed throughout the training and the test set, resulting in biased generalization estimates for ALC.

3.2.2. Proposed Dataset for Acoustic Location Classification

Our proposed dataset includes audio recordings of 8 different locations within the city of Graz, each assigned to one of the predefined acoustic scenes, i.e., *Public Square* and *Urban Green Space*. Thus, in Figure 3.1, we introduce the locations along with the corresponding acoustic scenes.
The dataset provides 300 s of audio data for each location per recording day, assembled on 3 consecutive working days (Wednesday, Thursday, Friday). Thus, we provide 7200 s of audio recordings, split into 720 10 s sequences, totalling approximately 700 MB.
To achieve a decent degree of temporal intraday identity, the daily recording time frame is restricted, starting at 9:00 am, and ending at 12:00 am. The movement of the recording device, i.e., the smartphone, is restricted to a minimum while recording. Thus, the smartphone is either placed on a stable surface, or held in the hand of the recording person. The position at the particular location remains throughout the recording sessions, except the position is occupied. In such case, the recording is conducted at the nearest possible position.
The dataset is assembled with our proposed Android application, utilizing a Nexus 6 as recording device. Thus, we provide 2 hours of monophonic audio recordings with a sample rate of 48 kHz and a bit depth of 16 bit.
To avoid any bias in the labelling process, we assign the locations of the dataset to the corresponding acoustic scene before the dataset is being analyzed. However, the distribution of locations between *Public Squares* and *Urban Green Spaces* represents a highly subjective task for humans.

For example, *Herz-Jesu-Kirche* shows characteristics of a *Public Square* during market hours, otherwise, the location tends to represent an *Urban Green Space*. Thus, we assign locations to the acoustic scene which fits best for the time being recorded, e.g., we declare *Herz-Jesu-Kirche* as *Urban Green Space*. Nevertheless, each location belongs to exactly one acoustic scene.
A similar ambivalence, w.r.t. market hours, is observed for *Kaiser-Josef-Platz*, however, due to local characteristics, *Kaiser-Josef-Platz* is declared as *Public Square* anyway.
Thus, the daily recording time frame enhances the temporal identity of locations, and hence simplifies the manual labelling of locations, w.r.t. acoustic scenes, to a certain degree.

Proposed Dataset		
Location	Acoustic Scene	Figure
Am Eisernen Tor	Public Square	3.17
Hauptplatz	Public Square	3.18
Jakominiplatz	Public Square	3.19
Kaiser-Josef-Platz	Public Square	3.20
Herz-Jesu-Kirche	Urban Green Space	3.21
Lessingstraße 25	Urban Green Space	3.22
Naglergasse 35	Urban Green Space	3.23
Stadtpark	Urban Green Space	3.24

Table 3.1.: Locations included in our proposed dataset along with the corresponding acoustic scenes.

Figure 3.17.: Am Eisernen Tor

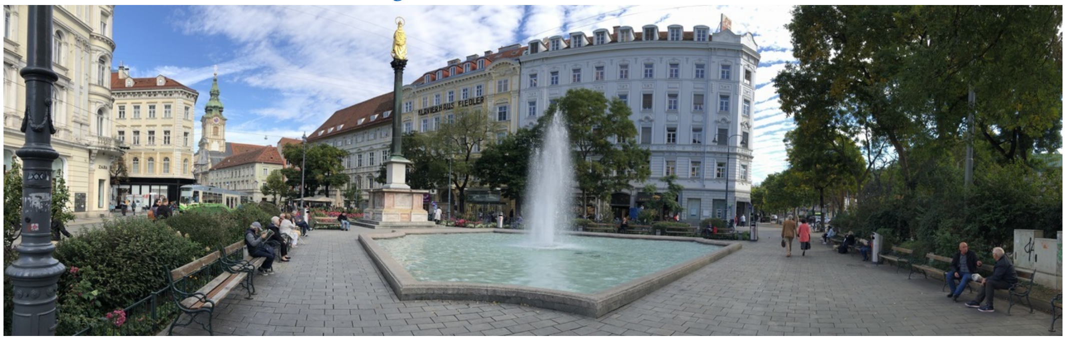


Figure 3.18.: Hauptplatz

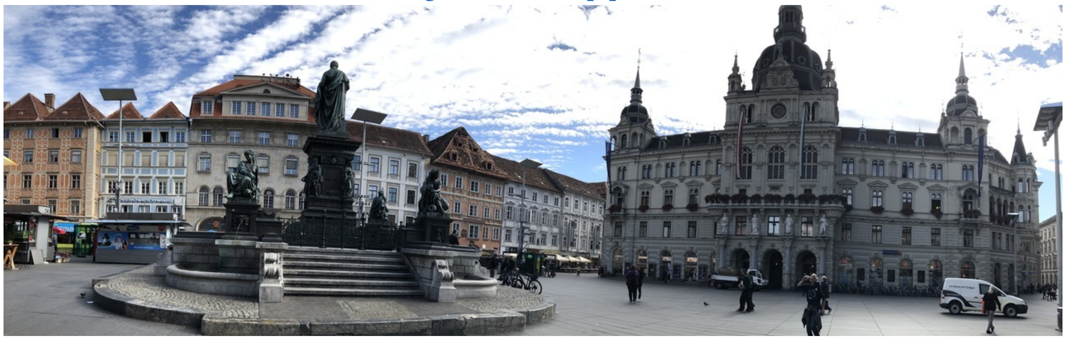


Figure 3.19.: Jakominiplatz

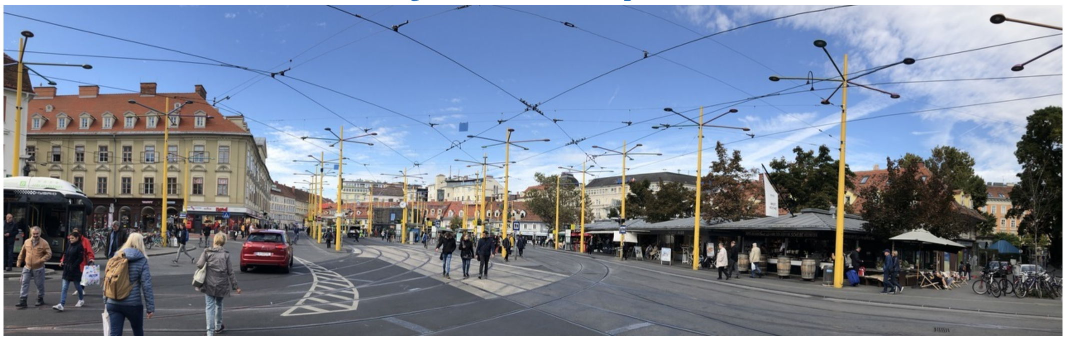


Figure 3.20.: Kaiser-Josef-Platz

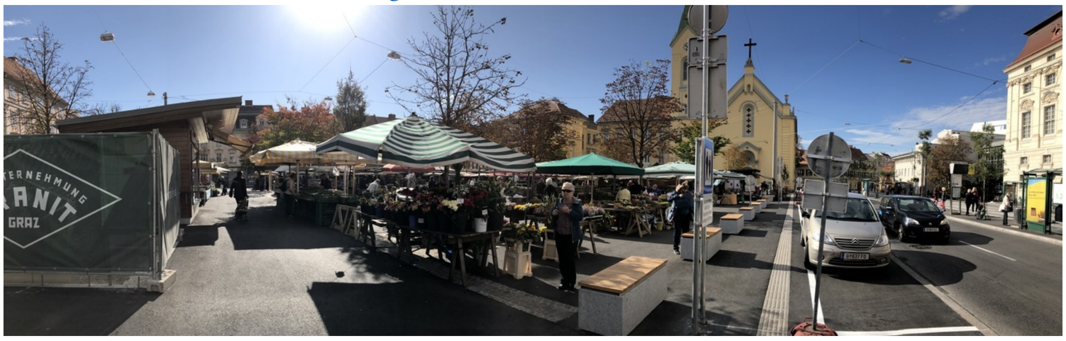


Figure 3.21.: Herz-Jesu-Kirche

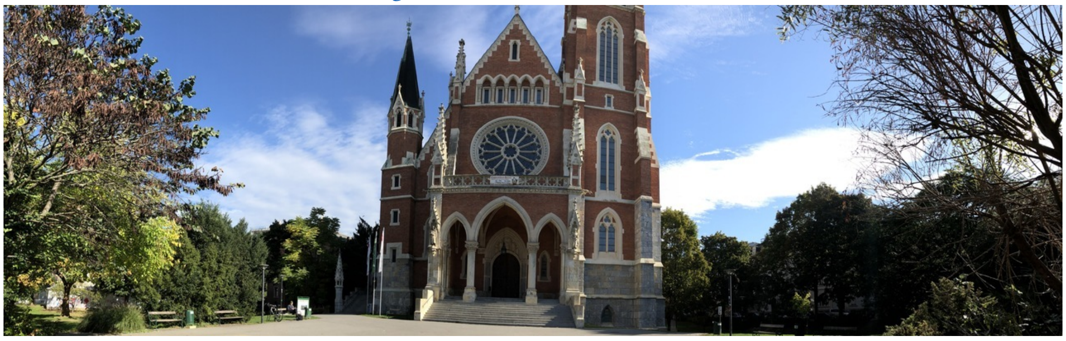


Figure 3.22.: Lessingstraße 25

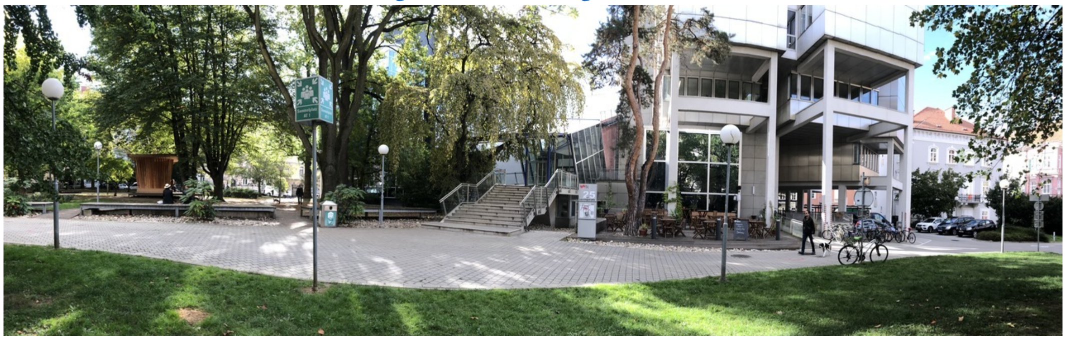


Figure 3.23.: Naglergasse 35

Figure 3.24.: Stadtpark

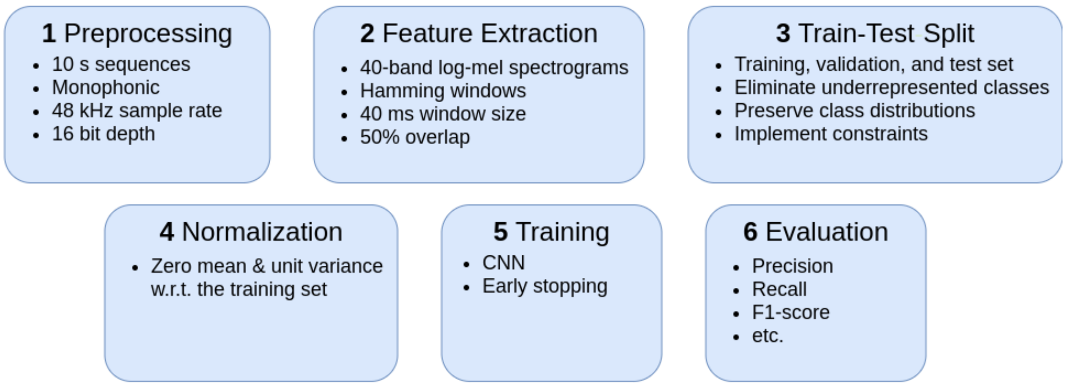
3.3. Classification Process

In the following, we provide an overview of the employed classification process. Thus, we introduce the applied preprocessing, along with the feature extraction, and the normalization methodology. Furthermore, we discuss the distribution of samples throughout the training, validation, and test set. Finally, we present the employed training procedure, and introduce the performance measures provided for the evaluation.
The preprocessing methodology depends on the underlying dataset, and hence differs between our proposed dataset, and the TAU development dataset.
The TAU development dataset provides stereophonic, more specifically, binaural, 10 s audio sequences with a sample rate of 48 kHz, and a bit depth of 24 bit. Thus,

to adapt the properties of the TAU development dataset to our proposed dataset, we transform the stereophonic audio data to monophonic signals by computing the arithmetic mean between the left and the right channel, and convert the bit depth from 24 bit to 16 bit.
Our proposed dataset provides monophonic audio recordings of variable length, featuring a sample rate of 48 kHz, and a bit depth of 16 bit. Hence, we split the audio recordings into 10 s sequences.
For the input of our CNN, i.e., the features, we extract 40-band log-mel spectrograms from the monophonic 10 s audio sequences by applying a Short-time Fourier Transform (STFT) with hamming windows, a window size of 40 ms, and an overlap of 50%, and transform the resulting power spectrogram into mel-scale. Thus, subsequently, in Section 3.3.1, we provide an introduction to the computation of log-mel spectrograms.
The split of the dataset into the training, validation, and test set depends on the particular experiment. However, in general, we split the dataset into the training and the test set, employing the majority for the training. Additionally, a small part of the training set is separated into the validation set, and hence employed to determine the model, i.e., the number of epochs, with the best generalization estimate, which is consequently utilized for the evaluation of the test set.
Furthermore, we uniformly distribute the samples, w.r.t. the targets, throughout the training, validation, and test set.
Additionally, to preserve an unbiased model, we eliminate underrepresented classes for particular experiments.
Finally, to investigate ALC, along with ASC, with temporal, and local constraints, we restrict the samples based on the date of recording, or the location of origin, to either the training, validation, or the test set. However, for the evaluation of ASC, the samples are not temporally, but locally, constrained (except for selected experiments), as approached in [Mesaros2018], and hence samples recorded at a later time, in comparison with samples located in the test set, might be distributed within the training set, and vice versa.
Thus, after splitting the dataset, we normalize the features to zero mean and unit variance by subtracting the mean, and dividing the features by the standard deviation of the training set.
Finally, we train the CNN on the samples provided by the training set, select the model with the best performance on the validation set, i.e., ES, and evaluate the selected model on the test set. Therefore, we additionally provide an in-depth introduction to the approached training procedure, along with the network architecture,

and hyperparameters, subsequently, in Section 3.3.2.
For the evaluation of our experiments, we provide the precision, recall, i.e., class-wise accuracy, F_1 score, and the number of supported samples within the test set. Additionally, we illustrate the confusion matrix, along with the normalized confusion matrix.
Thus, we provide sufficient information to investigate the practicality and limitations of ALC, along with ASC, on constrained training and test sets.
Summarized, we present the employed classification process in Figure 3.25.

Figure 3.25.: Overview of the employed classification process, thus, we start with the preprocessing to obtain monophonic 10 s audio sequences with a sample rate of 48 kHz and a bit depth of 16 bit, and proceed with the feature extraction by computing 40-band log-mel spectrograms. Next, we split the dataset into the training, validation, and the test set, and normalize the features to zero mean and unit variance w.r.t. the training set. Finally, we train the CNN on the training set, employ ES to determine the best performing model on the validation set, and provide the outcome of this model on the test set.



3.3.1. Feature Extraction

As input for the employed CNN, we utilize log-mel spectrograms, as proposed in [Valenti2016]. However, instead of 3 s sequences and 60 mel bands, we compute 40-band log-mel spectrograms of 10 s sequences, proposed as baseline for the DCASE 2019 ASC challenge in [Mesaros2018]. Thus, we utilize a window size of 40 ms with an overlap of 20 ms, i.e., 50%, based on hamming windows.

repetition? would suggest to re

Since the sequences of the TAU development dataset are recorded in stereo, more specifically, binaural, we first compute the monophonic signal by averaging both channels, i.e., computing the arithmetic mean, as approached for the TAU Urban Acoustic Scenes 2019 Mobile dataset (TAU mobile dataset), proposed in [Mesaros2018], to obtain comparable properties w.r.t. our proposed dataset.
First, to compute the log-mel spectrogram, a STFT is applied to the audio sequence. Thus, Figure 3.26 represents the monophonic amplitude of a 10 s sequence (*airport-london-6-275-a.wav*) of the TAU development dataset, and the resulting power spectrogram, after applying the STFT, is represented in Figure 3.27. Next, the frequencies are transformed into mel scale. Therefore, Figure 3.28 illustrates the corresponding mel filter bank. Finally, to compute the log-mel spectrogram, the logarithm is applied, and therefore the outcome is represented in Figure 3.29.
Summarized, we provide the computation of n-band log-mel spectrograms in Algorithm 4. Therefore, we utilize the libROSA[22]package to compute the STFT[23]and the mel filter banks[24]. *Additionally, we use the libROSA package to provide a numerically stable computation of the logarithm[25].
The final implementation is based on *melspectrogram*[26], provided by libROSA.

Algorithm 4 N-Band Log-Mel Spectrogram[22]

```
1: procedure FEATURES(y, sr=48000, windowSize=40, overlap=20, nMels=40)
2:     nFft ← int(sr × $\frac{1}{1000}$ × windowSize)          ▷ No. of samples per window
3:     hopLength ← int(sr × $\frac{1}{1000}$ × overlap) + 1 ▷ No. of overlapping samples - 1
4:     stft ← stft(y, nFft, hopLength, window=‘hamming’)       ▷ Compute STFT
5:     stft ← abs(stft)$^2$                                     ▷ Square absolut values
6:     melFilterBank ← mel(sr, nFft, nMels)                    ▷ Compute mel filter bank
7:     melSpectrogram ← melFilterBank · stft                   ▷ Apply dot product
8:     logMelSpectrogram ← 10 × $\log_{10}(\frac{\text{melSpectrogram}}{\max(\text{melSpectrogram})})$   ▷ Logarithmize*
9:     return logMelSpectrogram
```

[22]https://librosa.github.io (Accessed on: 2019-09-06)
[23]https://librosa.github.io/librosa/generated/librosa.core.stft.html (Accessed on: 2019-10-22)
[24]https://librosa.github.io/librosa/generated/librosa.filters.mel.html (Accessed on: 2019-10-22)
[25]https://librosa.github.io/librosa/generated/librosa.core.power_to_db.html (Accessed on: 2019-10-22)
[26]https://librosa.github.io/librosa/generated/librosa.feature.melspectrogram.html (Accessed on: 2019-10-22)

Figure 3.26.: Monophonic amplitude of a 10 s sequence (*airport-london-6-275-a.wav*) provided by the TAU development dataset.

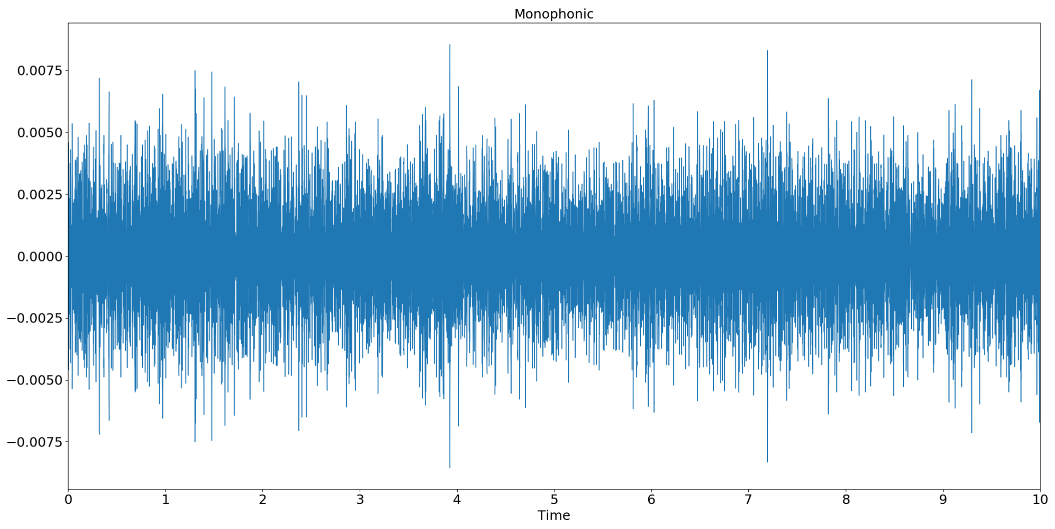


Figure 3.27.: Power spectrogram after applying the STFT based on *airport-london-6-275-a.wav*. To improve the representation, the result of the STFT is logarithmized.

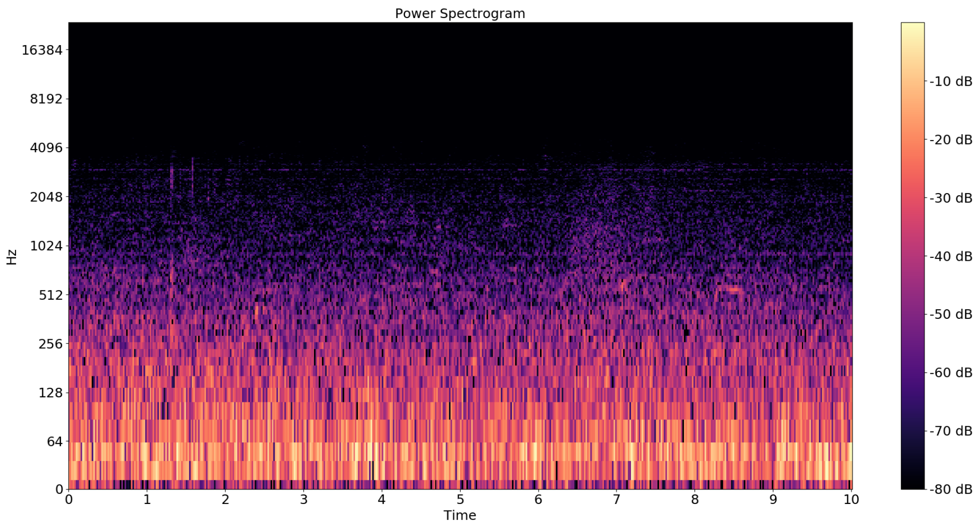


Figure 3.28.: 40-band mel filter bank to transform the result of the STFT of *airport-london-6-275-a.wav* into mel scale.

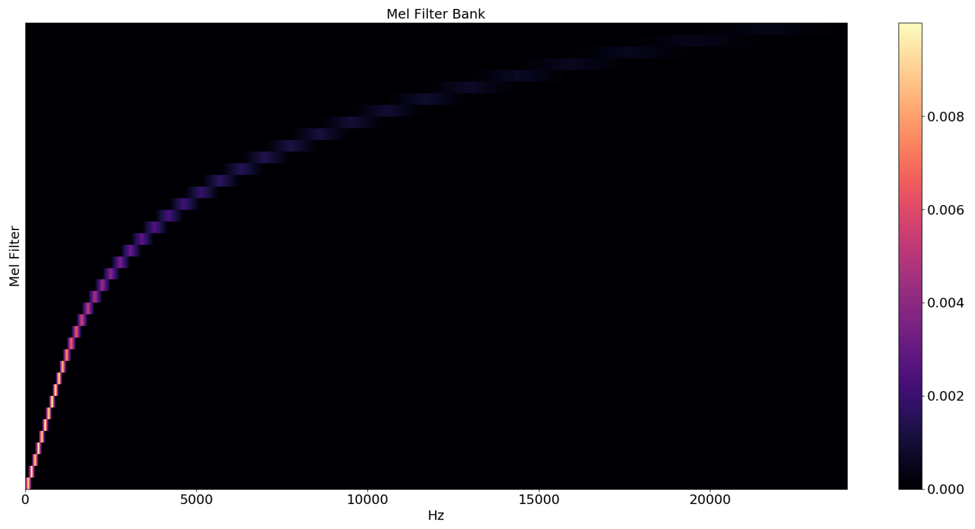
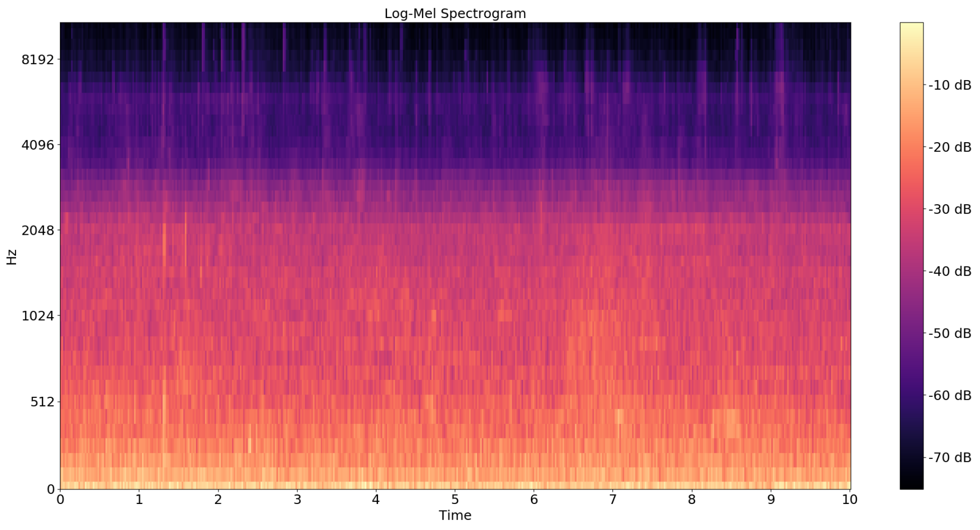


Figure 3.29.: The logarithmized mel spectrogram, i.e., 40-band log-mel spectrogram, representing the 10 s sequence *airport-london-6-275-a.wav* provided by the TAU development dataset.



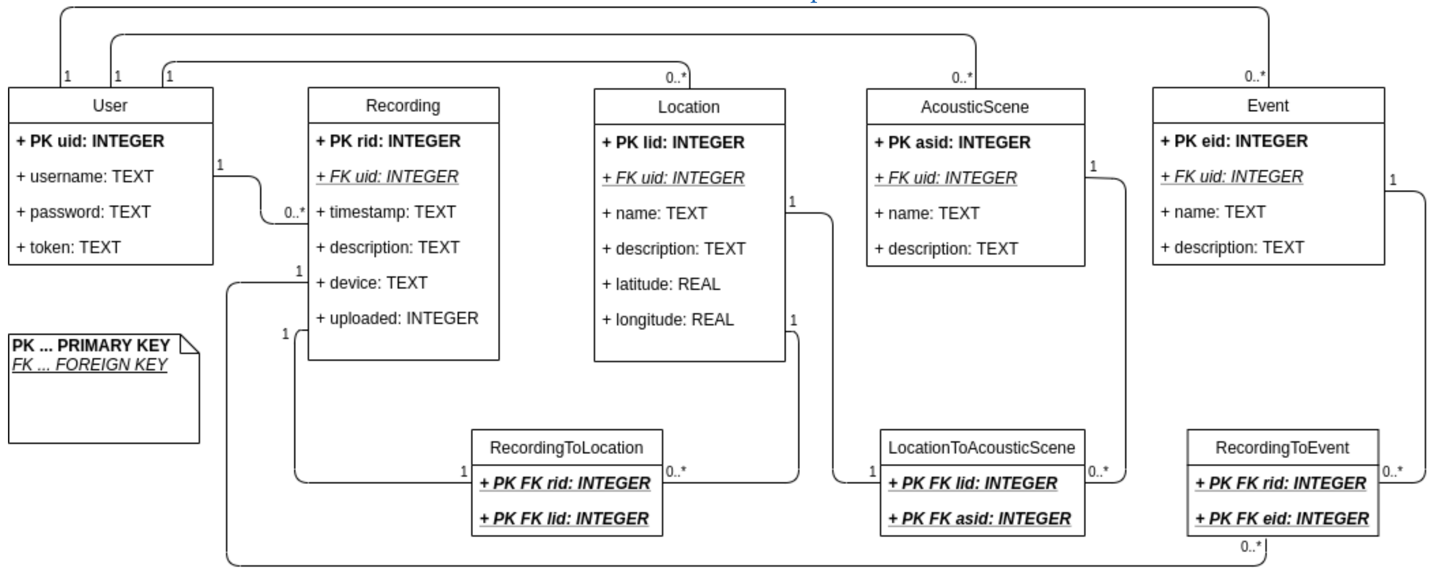
3.3.2. Architecture, Hyperparameters, and Training Procedure

For the architecture and the hyperparameters of our CNN, we follow the baseline approach for the DCASE 2019 ASC challenge proposed in [Mesaros2018].
Thus, for the first convolutional layer, we utilize 32 filters with a size of 7×7 and a stride of 1×1 with zero-padding. The first pooling layer applies max pooling with a pool size of 5×5. The second convolutional layer corresponds to the first, but utilizes 64 filters. Opposed to the first max pooling laying, the second pooling layer utilizes a pool size of 4×100. After the second pooling layer, the output is flattened and a fully connected layer with ReLU as activation function is introduced. Finally, a softmax layer is applied to compute the class-wise predictions.
Additionally, we utilize batch normalization before activating the output of the convolutional layers. Furthermore, 30% of the output neurons of the pooling layers, and of the fully connected layer, are dropped during training. The optimizer of choice is Adam with a learning rate of 0.001, and the loss function is realized by the categorical cross-entropy loss.
The other parameters, w.r.t. the Keras implementation, remain unchanged.
Summarized, we present the Keras layers in Figure 3.30, along with an illustration of the CNN in Figure 3.31.
The network is trained for 200 epochs with batches of 16 samples. Additionally, the training set is shuffled after each epoch. To avoid overfitting, and hence to improve generalization, we utilize ES. Thus, the model with the lowest validation loss, i.e., the CNN trained with the number of epochs providing the best generalization estimate, is selected and evaluated on the test set. Additionally, to avoid long runtimes for the network training, we interrupt the training after 20 stagnating epochs w.r.t. the validation loss.
The CNN is implemented in Python 3.7.3 with Keras 2.3.0, proposed in [Chollet2015], using TensorFlow-GPU 2.0.0, proposed in [Abadi2015], as backend.
For the training of the CNN, we utilize a GeForce GTX 1060 with CUDA 10.0 and cuDNN 7.6.2. Utilizing a GeForce GTX 1060 accelerates the training to 22 s per epoch, compared to 136 s with an Intel i7-6700HQ (4 cores, 8 threads, 2.6 GHz), w.r.t. ASC on the TAU development dataset.

Figure 3.30.: Implementation of the employed CNN represented by Keras layers.

```
Conv2D(filters=32, kernel_size=(7, 7), padding='same', input_shape=(40, 500, 1))
BatchNormalization()
Activation('relu')
MaxPool2D(pool_size=(5, 5))
Dropout(0.3)
Conv2D(filters=64, kernel_size=(7, 7), padding='same')
BatchNormalization()
Activation('relu')
MaxPool2D(pool_size=(4, 100))
Dropout(0.3)
Flatten()
Dense(units=100, activation='relu')
Dropout(0.3)
Dense(10, activation='softmax')
```

Figure 3.16.: ERD, illustrating the approached SQLite database schema. Due to the iterative development process, the final implementation differs from the illustrated database schema in minor aspects.



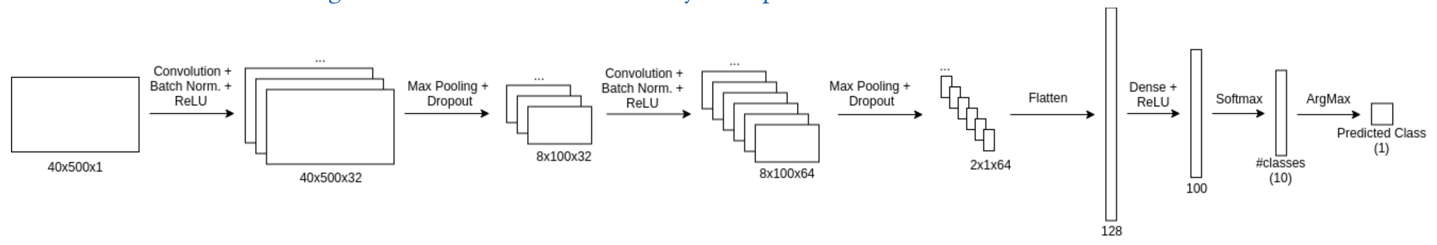


Figure 3.31.: Illustration of the 2-layer deep CNN utilized for our evaluation.

4. Results & Discussion

In this chapter, we provide the evaluation of our work. Thus, we introduce multiple experiments, and present the achieved outcomes of ASC, and ALC, employed on the TAU development dataset, introduced for the DCASE 2019 ASC challenge, and on our proposed dataset, especially established for the investigation of ALC. Furthermore, we provide an interpretation of the achieved results, discussing the practicality and limitations of ASC, and ALC, while focusing on local and temporal constraints on the distribution of samples throughout train-test splits.
Therefore, an overview of the conducted experiments is provided in Table 4.1, and the evaluated datasets, i.e., the TAU development dataset, and our proposed dataset, are specified in Table 4.2.
We start off with the Proof of Concept (PoC) of our CNN implementation on the TAU development dataset in Section 4.1. Therefore, we compare the results of our implementation to the published results[1] of the baseline system for the DCASE 2019 ASC challenge.
Next, in Section 4.2, we introduce an enhanced ES procedure, stopping the training once the error w.r.t. the validation set stops improving. Thus, we seek to improve the efficiency for the evaluation of multiple large models by introducing the possibility of stopping the training once the model has been converged, and hence avoid the maximum number of declared epochs.
The PoC evaluates the trained model on the test set, and the test set is additionally utilized to determine the best generalized model. Hence, this approach introduces a bias during the model selection w.r.t. the test set, thus, prevents a reliable evaluation. Unfortunately, over the period of this work, the DCASE 2019 challenge finished, and the targets of the TAU Urban Acoustic Scenes 2019, Evaluation dataset (TAU evaluation dataset) are not available to the public. Therefore, we utilize a part of the training set as validation set to employ ES, and hence restrict the test set to the

[1] `http://dcase.community/challenge2019/task-acoustic-scene-classification` (Accessed on: 2019-10-10)

evaluation, as presented in Section 4.3.
The suggested train-test split[2]restricts audio samples associated with a particular location to either the training or the test set. Furthermore, representatives of each city are distributed into the training, and the test set, respectively. However, for the remainder of this work, we do not intentionally enforce samples of each city to be present within the training as well as the test set.
For the application of ASC, we are encouraged to investigate the impact of locally constraining train-test splits. Thus, we additionally employ ASC without constraints on the TAU development dataset in Section 4.4.
Additionally, to provide a benchmark for the evaluation of ASC on our proposed dataset, we present the results of employing ASC with local constraints on the TAU development dataset, including only samples associated with *Urban Parks* and *Public Squares*, in Section 4.5.
Finally, for the TAU development dataset, we present the results of employing ALC on locations labeled as *Urban Park* in Section 4.6.
Next, we introduce our proposed dataset, especially established to investigate the impact of temporal constraints on ALC.
We start off with the evaluation of ASC without constraints on the train-test split in Section 4.7.
Additionally, we provide the evaluation of ASC on our proposed dataset, implementing the approach of the suggested train-test split proposed for the DCASE 2019 ASC challenge, i.e., introducing local constraints, in Section 4.8.
Furthermore, in consideration of ASC on our proposed dataset, we substitute the local constraints by temporal constraints. Thus, the audio samples of the training and the test set constitute different recording dates, as presented in Section 4.9.
Finally, we proceed with the evaluation of ALC on our proposed dataset. Thus, in Section 4.10, we employ ALC without constraints on the train-test split, and provide the evaluation of the proposed temporal constraints on ALC in Section 4.11.

[2]http://dcase.community/challenge2019/task-acoustic-scene-classification (Accessed on: 2019-10-10)

Table 4.1.: Overview of the conducted experiments for the evaluation of ASC and ALC on the TAU development dataset and our proposed dataset.

Experiments	
TAU Acoustic Scene Classification Proof of Concept	4.1
TAU Acoustic Scene Classification Early Stopping	4.2
TAU Acoustic Scene Classification	4.3
TAU Acoustic Scene Classification Unconstrained	4.4
TAU Acoustic Scene Classification on Urban Parks & Public Squares	4.5
TAU Acoustic Location Classification on Urban Parks	4.6
Proposed Acoustic Scene Classification	4.7
Proposed Acoustic Scene Classification with Local Constraints	4.8
Proposed Acoustic Scene Classification with Temporal Constraints	4.9
Proposed Acoustic Location Classification	4.10
Proposed Acoustic Location Classification with Temporal Constraints	4.11

4.1. TAU Acoustic Scene Classification Proof of Concept

In the following, we present the results of our CNN, employed for ASC on the TAU development dataset. Thus, we provide a PoC for the employed preprocessing, classification, and evaluation methodology.
Summarized, we employ a 2-layer deep CNN to classify 40-band log-mel spectrograms based on 10 s audio samples to the corresponding acoustic scene, i.e., an abstraction of the location w.r.t. the acoustic environment where the recording is conducted.
First, to adapt the characteristics of the TAU development dataset to correspond to our proposed dataset, the stereophonic 10 s audio samples with a bit depth of 24 bit are converted to monophonic signals with a bit depth of 16 bit.
Additionally to the official DCASE 2019 baseline, we enhance the employed ES methodology. Thus, the training is stopped once the error w.r.t. the validation set, i.e., test set, stops improving for at least 20 epochs.
Furthermore, we utilize the suggested train-test split proposed for the DCASE 2019 ASC challenge, i.e., restricting samples based on the associated location to either the training or the test set.

Table 4.2.: Properties of the TAU development dataset and our proposed dataset w.r.t. the number of samples and recordings, the duration of recordings, and the number of featured acoustic scenes and locations.

	TAU development dataset[3]	Proposed dataset
Number of 10 s samples	14400	720
Number of recordings	~ 1000	24
Duration of one recording	~ 144	~ 300
Acoustic scenes	10	2
Locations	514	8

Thus, the network is trained on the training set until convergence, or for a maximum of 200 epochs, and the most generalized model, i.e., the model trained until the epoch resulting in the lowest test error, is utilized for the evaluation.
Therefore, in Table 4.3, we present the published class-wise accuracies for the DCASE 2019 ASC challenge, along with the class-wise accuracies of our implementation, representing the arithmetic mean over 10 independent training runs. Additionally, precision, F_1 score, and the number of samples for each class of the test set are provided. Thus, we observe a minor difference of 0.66%, based on the 62.51% of the published results, and 63.17% of our implementation.
This deviation might be deduced from the reduced bit depth, the enhanced ES methodology, and the randomness in the parameter initialization. Furthermore, utilizing GPUs for training NNs introduces non-determinism[4], resulting in slight deviations as well.
Additionally, we illustrate precision, recall, and F_1 score for each acoustic scene in Figure 4.1.
Thus, Figure 4.1 shows that the results remain unbiased w.r.t. underrepresented classes, and overrepresented classes, respectively.
Furthermore, the confusion matrix, along with the normalized confusion matrix, are represented in Figure A.1, and Figure 4.2, respectively.
As illustrated, certain acoustic scenes are harder to classify than others, e.g., *Airport* is classified correctly by 53%, opposed to *Urban Park*, reaching an accuracy of 83%. Furthermore, certain acoustic scene pairs appear hard to be differentiated, e.g., *Airport* and *Shopping Mall*, resulting in a misclassification of *Airport* to *Shopping*

[4]http://dcase.community/challenge2019/task-acoustic-scene-classification (Accessed on: 2019-09-23)

Mall by 32%.

Table 4.3.: Published class-wise accuracies of employing ASC on the TAU development dataset, along with arithmetic mean of the class-wise accuracies w.r.t. our implemented PoC, based on the suggested train-test split, and averaged over 10 independent training runs. Additionally, precision, F_1 score, and the number of samples for each class of the test set are provided.

Acoustic Scenes	*Accuracy Official*[5]	Accuracy	Precision	F_1 score	Support
Airport	48.40%	53.35%	64.63%	58.33%	421
Bus	62.30%	71.37%	64.73%	67.78%	415
Metro	65.10%	58.87%	65.58%	61.89%	433
Metro station	54.50%	50.14%	61.97%	54.95%	435
Park	83.10%	82.59%	87.02%	84.67%	386
Public square	40.70%	44.73%	44.75%	44.52%	387
Shopping mall	59.40%	68.37%	54.68%	60.64%	441
Street pedestrian	60.90%	61.54%	54.53%	57.51%	429
Street traffic	86.70%	81.59%	84.00%	82.68%	402
Tram	64.00%	59.20%	58.31%	58.52%	436
Average	62.51%	63.17%	64.02%	63.15%	
+/-	0.60%	1.23%	1.44%	1.28%	

Figure 4.1.: Arithmetic mean of precision, recall, and F_1 score of employing our ASC PoC on the suggested train-test split of the TAU development dataset, averaged over 10 independent training runs.

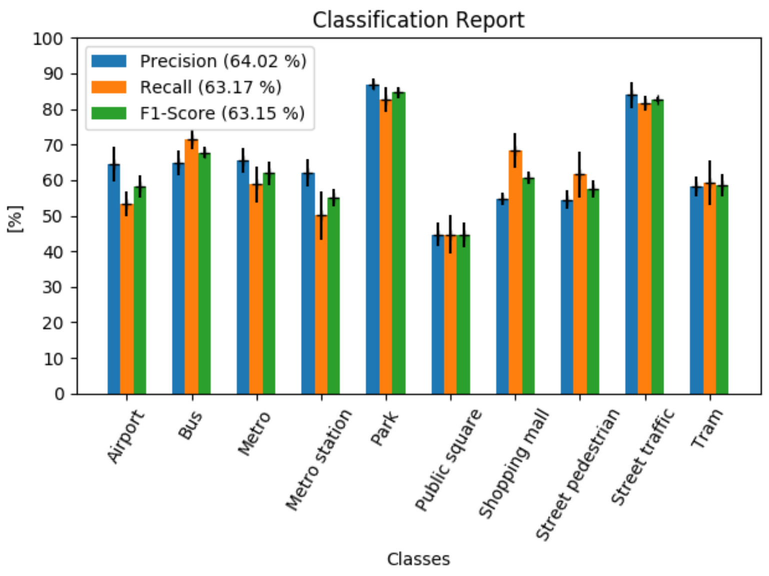
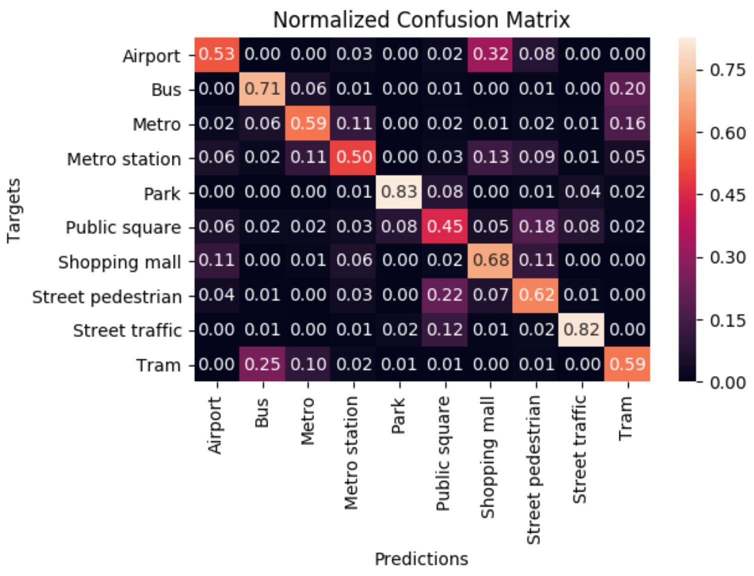


Figure 4.2.: Normalized confusion matrix, illustrating the arithmetic mean, w.r.t. the class-wise accuracies, of employing ASC on the suggested train-test split for the TAU development dataset, averaged over 10 independent training runs.



4.2. TAU Acoustic Scene Classification Early Stopping

In the following, we provide the results of the enhanced ES approach, introduced for our employed training procedure.
Opposed to the official DCASE 2019 ASC baseline, we interrupt the training after 20 epochs of stagnating validation loss. However, we keep a maximum number of 200 epochs for the training.
Thus, we monitor the network on the validation set, i.e., on the test set, while training the network on the training set. Furthermore, we store the current model, i.e., the model parameters, every time the validation loss improves. Finally, we evaluate the model with the lowest validation loss on the test set, i.e., on the evaluation set. As represented in Figure 4.3, the performance of the model on unseen samples remains constant, or even becomes worse, as soon as the validation loss stops decreasing for a reasonable number of, e.g., 20, epochs. Thus, we achieve satisfying

results, while keeping the number of training epochs comparatively low, as represented in Figure 4.4.
However, we cannot completely eliminate the risk of missing a model, i.e., an epoch, with a lower validation loss within 200 epochs.

Figure 4.3.: Loss and accuracy of the training and validation set of employing our PoC w.r.t. ASC on the suggested train-test split of the TAU development dataset until 200 epochs.

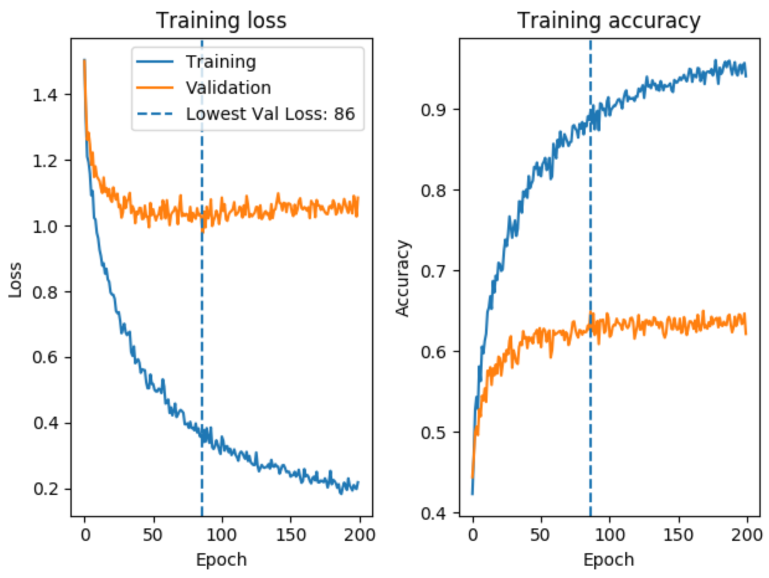
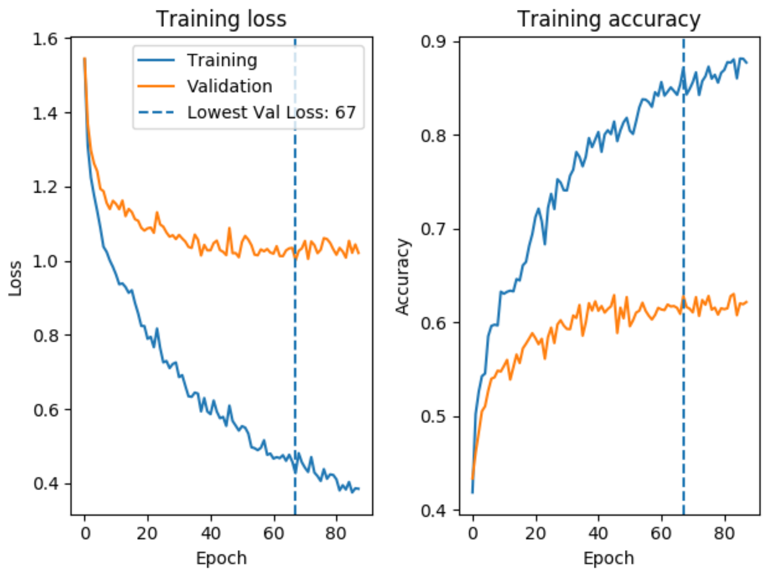


Figure 4.4.: Training and validation loss, and accuracy, respectively, of our PoC employed for ASC on the suggested train-test split of the TAU development dataset, while stopping the training after 20 epochs of non-improvement w.r.t. the validation loss.



4.3. TAU Acoustic Scene Classification

For the PoC of our CNN implementation, previously presented in Section 4.1, we employ ASC on the suggested train-test split, proposed for the DCASE 2019 ASC challenge. Thus, we utilize the test set to determine the model with the lowest validation loss and receive a performance estimate for the test set. Therefore, the test set is not only utilized to evaluate the performance, but is also involved in the training process, more precisely, in the model selection.
Since the DCASE 2019 ASC challenge expired during this thesis, and the targets for the TAU evaluation dataset are not provided to the public, we are restricted to the TAU development dataset.
Thus, to achieve unbiased results w.r.t. the test set, we utilize 10% of the training set as separated validation set. Additionally, the validation set is assembled to restrict audio samples associated with different locations to either the training or the validation set.

In Table 4.4, we provide the class-wise accuracies, along with precision, and F_1 score, averaged over 10 independent training runs, as well as the number of represented samples for each acoustic scene of the test set.
As illustrated, the arithmetic mean of the class-wise accuracy over 10 independent training runs drops from 63.17% to 57.19%, compared to the PoC.
This decrease in the accuracy of $5,98\%$ can be explained by the decreased number of available training samples, as well as on the strict separation between the validation and the test set.
Additionally, we illustrate precision, recall, and F_1 score, for each acoustic scene, in Figure 4.5. Finally, we represent the confusion matrix in Figure A.2, along with the normalized confusion matrix in Figure 4.6.

Table 4.4.: Arithmetic mean of the class-wise accuracies, precision, and F_1 score, averaged over 10 independent training runs, along with the number of samples for each class, of employing ASC on the TAU development dataset and utilizing 10% of the training set as separated validation set. Additionally, the arithmetic mean of the class-wise accuracies for ASC on the TAU development dataset based on the suggested train-test split, i.e., the outcomes of our PoC, are presented.

Acoustic Scenes	*Accuracy PoC*	Accuracy	Precision	F_1 score	Support
Airport	53.35%	41.69%	58.05%	46.98%	421
Bus	71.37%	51.42%	56.47%	53.04%	415
Metro	58.87%	60.44%	56.42%	57.90%	433
Metro station	50.14%	40.32%	59.90%	47.79%	435
Park	82.59%	81.76%	85.15%	83.23%	386
Public square	44.73%	35.48%	40.82%	37.71%	387
Shopping mall	68.37%	64.78%	52.19%	56.89%	441
Street pedestrian	61.54%	60.40%	50.53%	54.62%	429
Street traffic	81.59%	81.72%	78.75%	80.09%	402
Tram	59.20%	53.88%	45.85%	49.11%	436
Average	63.17%	57.19%	58.41%	56.74%	
+/-	1.23%	2.30%	1.91%	2.46%	

Figure 4.5.: Arithmetic mean of precision, recall, and F_1 score, averaged over 10 independent training runs, of employing ASC on the TAU development dataset while utilizing 10% of the training set as separated validation set.

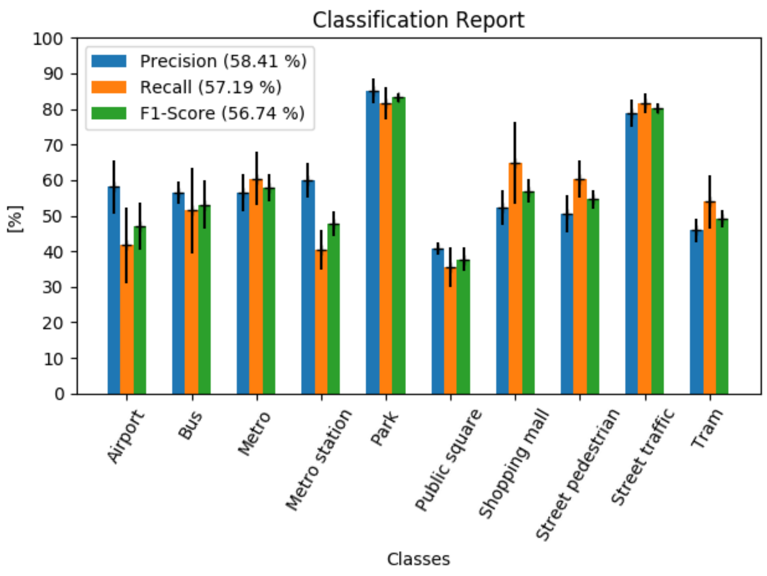
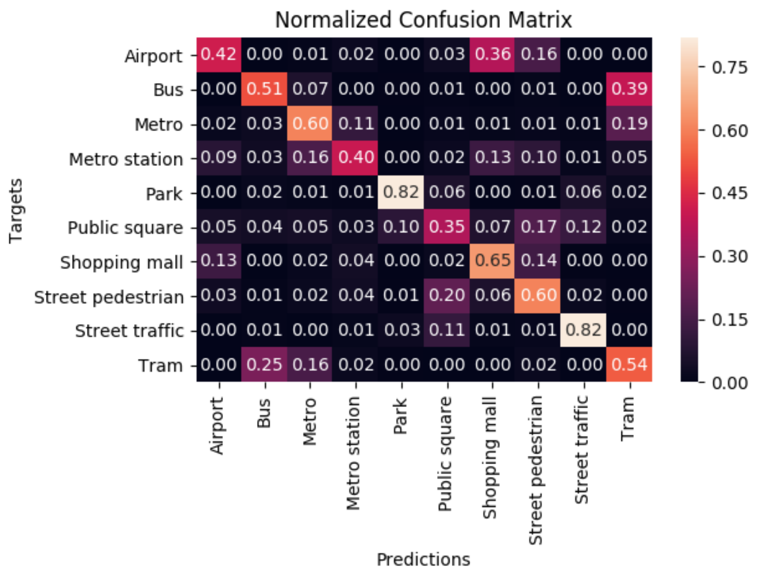


Figure 4.6.: Normalized confusion matrix, representing the arithmetic mean of the class-wise accuracies, averaged over 10 independent training runs, of employing ASC on the TAU development dataset with a separated validation set, i.e., utilizing 10% of the training set exclusively for validation.



4.4. TAU Acoustic Scene Classification Unconstrained

Next, to investigate the impact of restricting samples based on the associated location to either the training, validation, or test set, previously presented in Section 4.3, we provide the results of employing ASC on the TAU development dataset without the suggested train-test split.

Thus, rather than restricting samples to one set w.r.t. the labelled location, we randomly split the TAU development dataset into a training set (70%) and a test set (30%). Additionally, to employ ES, and hence to select the model with the highest generalization score, we split 10% of the training set into a separated validation set. To avoid under- and overrepresented classes, although randomly split, we ensure to uniformly distribute, w.r.t. acoustic scenes, the samples throughout the training, validation, and test set.

Thus, we present the results of employing ASC without constraints on the train-test split on the TAU development dataset in Table 4.5, providing the class-wise accuracies averaged over 10 independent training runs.
Additionally, for each of the 10 training runs, a new random split is employed, and the training is stopped after 20 epochs of non-improvement w.r.t. the validation loss.
As shown in Table 4.5, the arithmetic mean of the class-wise accuracy, averaged over 10 independent training runs, improved from 57.19% to 76.45%.
This increase of the accuracy by 19.26% indicates the importance of reasonable constraints on the train-test split. Otherwise, the classification accuracy does not express the actual classification performance w.r.t. generalization. [Valenti2016]
Additionally, precision, recall, and F_1 score, for each class, are illustrated in Figure 4.7. Finally, the confusion matrix, along with the normalized confusion matrix, of employing unconstrained ASC on the TAU development dataset, are represented in Figure A.3, and Figure 4.8, respectively.

Figure 4.7.: Arithmetic mean of precision, recall, and F_1 score, averaged over 10 independent training runs, of employing ASC on the TAU development dataset without restricting samples based on the associated location to either the training or the test set.

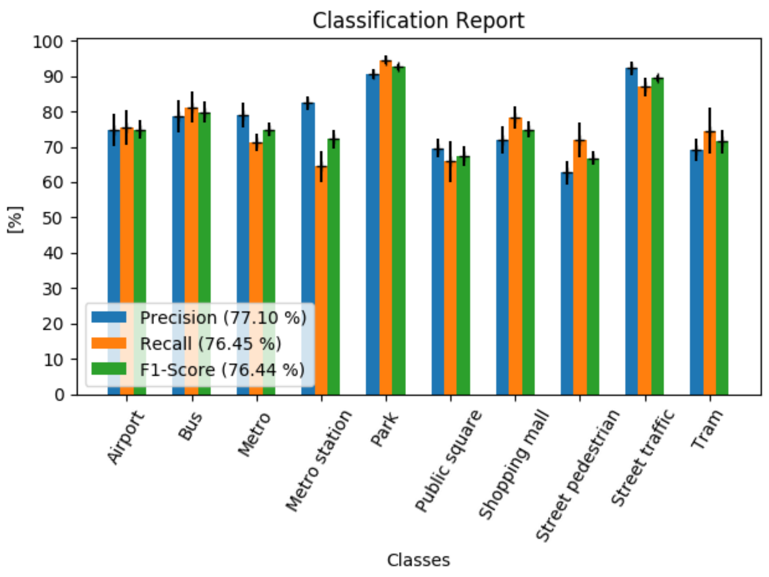
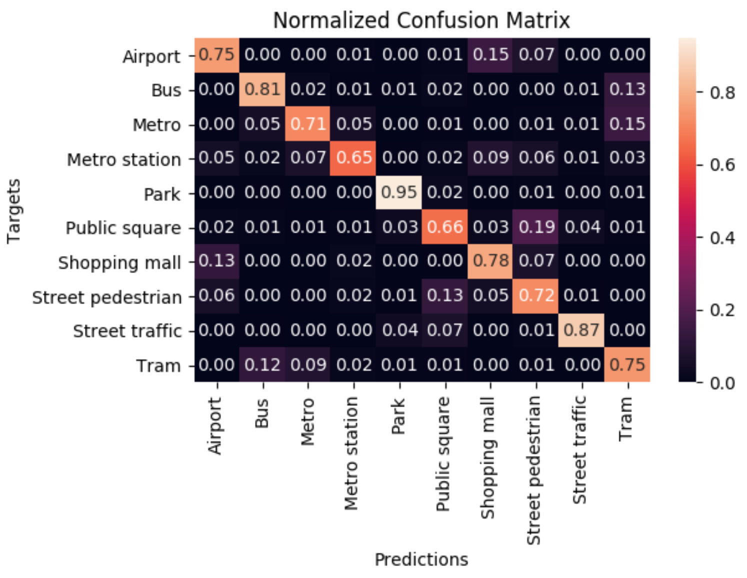


Table 4.5.: Arithmetic mean of the class-wise accuracies, along with precision, and F_1 score, averaged over 10 independent training runs, and the number of samples representing the corresponding acoustic scene within in test set, providing the outcomes of employing ASC on the TAU development dataset while omitting the restrictions of the suggested train-test split.

Acoustic Scenes	*Acc. Locally Constrained*	Accuracy	Precision	F_1 score	Support
Airport	41.69%	75.42%	74.83%	74.91%	432
Bus	51.42%	81.18%	78.70%	79.78%	432
Metro	60.44%	71.25%	78.99%	74.85%	432
Metro station	40.32%	64.54%	82.37%	72.27%	432
Park	81.76%	94.56%	90.60%	92.53%	432
Public square	35.48%	65.90%	69.55%	67.50%	432
Shopping mall	64.78%	78.17%	71.93%	74.82%	432
Street pedestrian	60.40%	72.01%	62.70%	66.83%	432
Street traffic	81.72%	86.90%	92.16%	89.40%	432
Tram	53.88%	74.54%	69.15%	71.56%	432
Average	57.19%	76.45%	77.10%	76.44%	
+/-	2.30%	1.78%	1.65%	1.78%	

Figure 4.8.: Normalized confusion matrix, representing the arithmetic mean of the class-wise accuracies, averaged over 10 independent training runs, of employing ASC on the TAU development dataset, leaving out the constraints of the suggested train-test split.



4.5. TAU Acoustic Scene Classification on Urban Parks & Public Squares

Additionally, to obtain comparable results for the evaluation of our proposed dataset, we present the results of employing ASC on the TAU development dataset, only including samples associated with *Urban Parks* and *Public Squares*. Since our proposed dataset provides samples within 2 acoustic scenes, *Urban Green Space* and *Public Square*, which relate to the acoustic scenes *Urban Park* and *Public Square*, represented within the TAU development dataset, we provide a benchmark w.r.t. ASC for our proposed dataset.
Thus, we utilize the suggested train-test split for the TAU development dataset, but solely include samples labelled as *Urban Park* or *Public Square*. Additionally, we employ 10% of the training set for validation, hence, to determine the model with the highest generalization estimate, and to stop the training after the validation loss suspends for at least 20 epochs. Furthermore, we maintain the restriction for samples associated with the same location to be either placed in the training or the validation set.
Therefore, we present the class-wise accuracies, along with precision, F_1 score, and the number of samples for each class, of employing ASC on *Urban Parks* and *Public Squares* of the TAU development dataset, averaged over 10 independent training runs, in Table 4.6.
Thus, we achieve a class-wise accuracy of 89.45% for the classification between *Urban Parks* and *Public Squares* of the TAU development dataset.
Additionally, precision, recall, and F_1 score, for each class, i.e., acoustic scene, are illustrated in Figure 4.9. Finally, we present the confusion matrix, along with the normalized confusion matrix, in Figure A.4, and Figure 4.10, respectively.

Table 4.6.: Arithmetic mean of the class-wise accuracies, precision, and F_1 score, averaged over 10 independent training runs, along with the number of samples per class represented within the test set, of employing ASC on the TAU development dataset limited to *Urban Parks* and *Public Squares*.

Acoustic Scenes	Accuracy	Precision	F_1 score	Support
Park	87.82%	90.89%	89.27%	386
Public square	91.09%	88.31%	89.62%	387
Average	89.45%	89.60%	89.45%	
+/-	0.51%	0.60%	0.51%	

Figure 4.9.: Arithmetic mean of precision, recall, and F_1 score, averaged over 10 independent training runs, of employing ASC on the TAU development dataset, considering solely locations associated with *Urban Parks* and *Public Squares*.

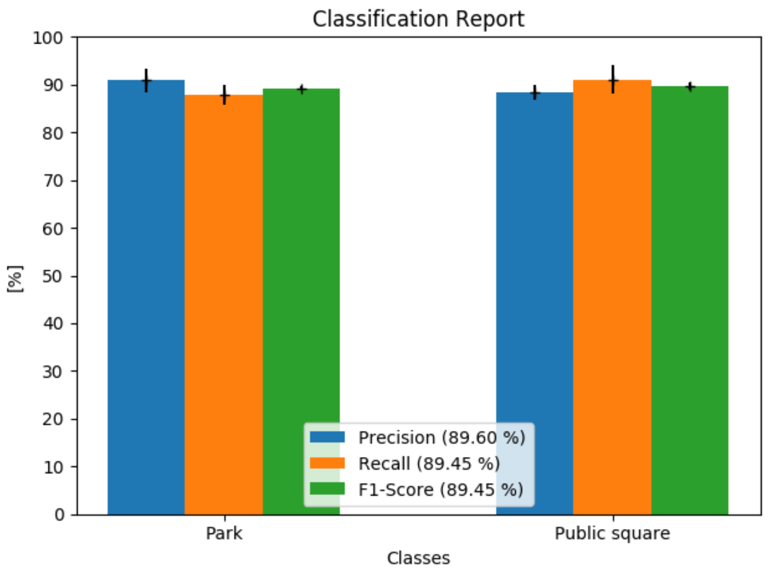
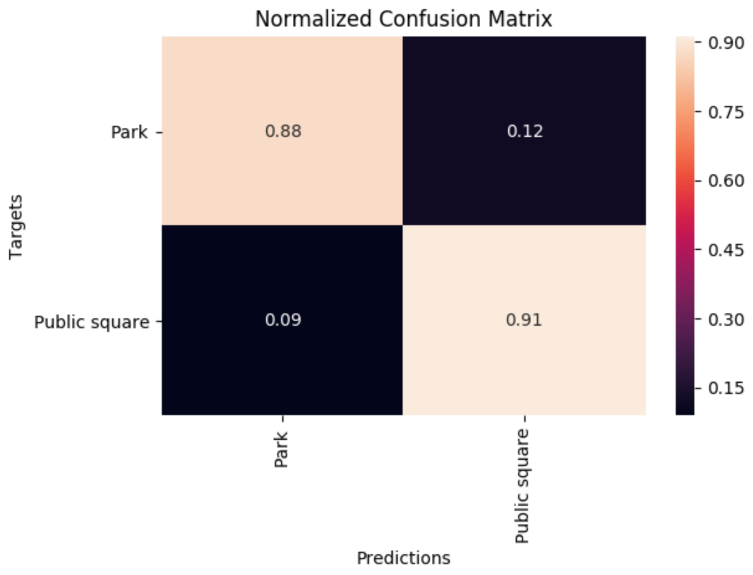


Figure 4.10.: Normalized confusion matrix, illustrating the arithmetic mean of the class-wise accuracies, averaged over 10 independent training runs, of employing ASC on locations labelled as *Urban Park* or *Public Square* within the TAU development dataset.



4.5.1. Transfer TAU Acoustic Scene Classification

To investigate the capability of transferring models between related datasets, we evaluate the CNN trained on the acoustic scenes *Urban Park* and *Public Square* of the TAU development dataset on our proposed dataset. Hence, the acoustic scene *Urban Park* corresponds to *Urban Green Space*, and the acoustic scene *Public Square* is preserved for the evaluation on our proposed dataset.
As presented in Table 4.7, the transferred model, trained on the particular acoustic scenes of the TAU development dataset, yields an average accuracy of 93.00%, and a standard deviation of 1.51%, when evaluated on our proposed dataset.
Since the transferred model implies local and temporal constraints for the evaluation, and the audio samples of our proposed dataset are recorded with a smartphone, opposed to a professional audio recorder, employed for the samples of the TAU development dataset, the resulting accuracies appear relatively high compared to the results when trained and evaluated on our proposed dataset, as presented in Table 4.8.

Therefore, the transferred model loses 5.74%, and 2.14%, compared to unconstrained, and temporally constrained ASC, respectively, but gains 4.85% compared to locally constrained ASC.
However, as the size of our proposed dataset is comparatively small, opposed to the TAU development dataset, a generalization to acoustic scenes becomes difficult when introducing local constraints, and hence yields relatively low classification accuracies when trained on our proposed dataset.
Thus, the outcomes emphasize the benefits of transferring trained models when dealing with related datasets.

Table 4.7.: Arithmetic mean of the class-wise accuracies, precision, F_1 score, and the number of evaluated samples provided by our proposed dataset, averaged over the 10 models trained on the acoustic scenes *Urban Park* and *Public Square* of the TAU development dataset.

Acoustic Scenes	Accuracy	Precision	F_1 score	Support
Park	93.64%	92.52%	93.05%	360
Public Square	92.36%	93.59%	92.94%	360
Average	93.00%	93.05%	93.00%	
+/-	1.51%	1.47%	1.51%	

Table 4.8.: Outcomes of the ASC experiments evaluated on our proposed dataset.

Experiment	Accuracy	Reference
TAU ASC Transferred	93.00%	4.7
Proposed ASC without Constraints	98.74%	4.10
Proposed ASC with Local Constraints	88.15%	4.12
Proposed ASC with Temporal Constraints	95.14%	4.18

4.6. TAU Acoustic Location Classification on Urban Parks

In the following, we present the results of employing ALC on the TAU development dataset. More specifically, we employ ALC on the locations labelled as *Urban Park*, constituting a classification task between 41 locations.
Due to the limited number of 8 locations provided by our proposed dataset, and the number of classes, i.e., 10 acoustic scenes, which the CNN is optimized for, we solely include locations labelled as *Urban Park*, rather than the whole TAU development dataset, hence, keeping the number of classes, i.e., locations, reasonably low. Furthermore, we provide a benchmark for employing ALC on our proposed dataset.
Thus, we randomly split the samples within the acoustic scene *Urban Park* into a training set (70%) and a test set (30%). Furthermore, to enable ES, and hence to select the best performing model, i.e., the model trained until the epoch resulting in the lowest validation loss, we utilize 10% of the training set as validation set. Besides, the training is stopped once the validation loss refuses to improve for at least 20 epochs. Additionally, we ensure that the targets are uniformly distributed w.r.t. the locations throughout the training, validation, and test set. Furthermore, to avoid underrepresented classes, we remove locations that support less than 20 samples. To enhance the reproducibility of the results, we conduct 10 independent training runs and employ a new random split for each run.
Thus, in Table 4.9, we present the arithmetic mean of the class-wise accuracies averaged over 10 independent training runs, along with precision and F_1 score. Hence, we achieve an averaged class-wise accuracy of 84.27% when employing ALC on the TAU development dataset limited to *Urban Parks*.
Besides, the number of samples per class provided by the test set, i.e., the support, might differ by one sample between the training runs based on the utilized train-test split implementation.
Since we aim to classify audio samples to the corresponding location of origin, restricting each location to either the training or the test set, as approached for ASC, becomes infeasible. Thus, *samples originated in the same recording, or a very similar recording w.r.t. the position at the location and the time frame, are distributed throughout the training and the test set.* Therefore, employing ALC, without introducing a significant temporal gap between training and test samples, draws a tampered classification estimate w.r.t. generalization [Valenti2016], and results in

an overfitted model [Mesaros2018a], potentially unable to perform comparatively well on unseen samples, i.e., on samples recorded on a different date.
Additionally, in Figure 4.11, precision, recall, and F_1 score, are illustrated. Finally, the normalized confusion matrix is represented in Figure 4.12.

Table 4.9.: Arithmetic mean of the class-wise accuracies of employing ALC on the TAU development dataset, averaged over 10 independent training runs, along with precision and F_1 score, illustrating the classification performance between 41 locations, each represented by the acoustic scene *Urban Park*.

	Accuracy	Precision	F_1 score
Average	84.27%	85.56%	84.03%
+/-	1.75%	1.60%	1.78%

Figure 4.11.: Arithmetic mean of precision, recall, and F_1 score of employing ALC on locations labelled as *Urban Park* of the TAU development dataset, averaged over 10 independent training runs.

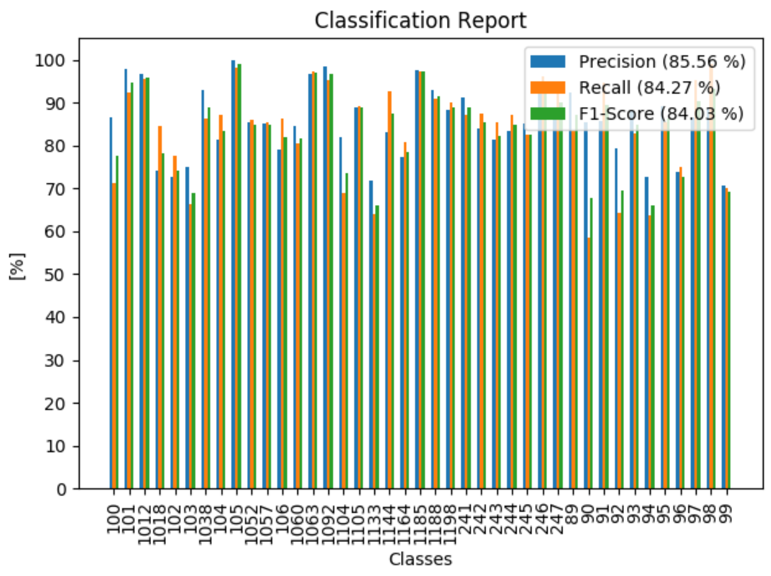
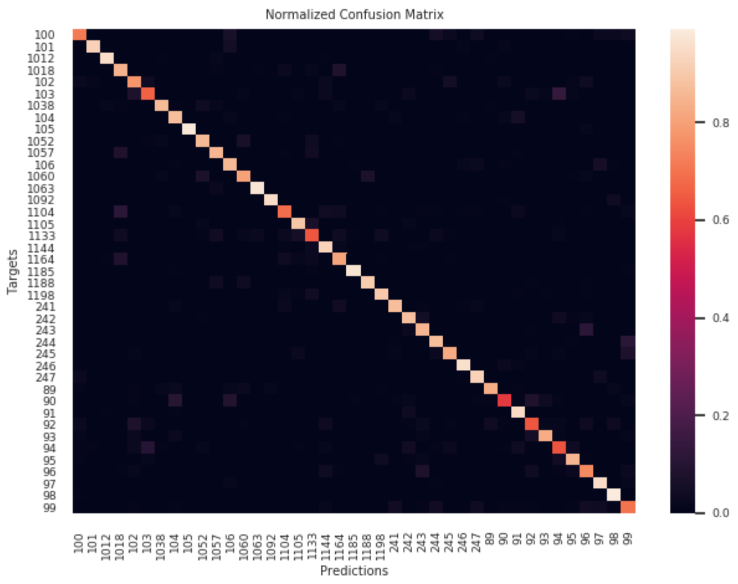


Figure 4.12.: Normalized confusion matrix, representing the arithmetic mean of the class-wise accuracies of employing ALC on locations labelled as *Urban Park* of the TAU development dataset, averaged over 10 independent training runs.



4.7. Proposed Acoustic Scene Classification

To introduce our proposed dataset, represented by 2 acoustics scenes, and 8 locations, respectively, and to provide a benchmark for locally and temporally constrained ASC, we provide the evaluation of ASC without constraints on the train-test split.
Thus, to enhance the comparability to temporally constrained ASC, subsequently presented in Section 4.9, we split the dataset into 3 equally sized folds—the provided samples are recorded on 3 consecutive days and introducing temporal constraints results in the stated distribution.
More specifically, we split the dataset in the training set (66%) and in the test set (33%). Additionally, we utilize 50% of the training set as validation set to enable ES, and hence to determine the best trained model. Furthermore, the training is stopped after 20 epochs of non-improvement w.r.t. the validation loss.
For the evaluation, we conduct 10 independent training runs and randomly split

the dataset in the training, validation, and the test set for each training run independently. Additionally, the samples are uniformly distributed w.r.t. the acoustic scenes throughout the sets.
Thus, in Table 4.10, we present the arithmetic mean of the class-wise accuracies averaged over 10 independent training runs, as well as precision, F_1 score, and the number of samples for each acoustic scene of the test set.
Hence, we observe an accuracy of 98.74% for employing ASC on *Public Squares* and *Urban Green Spaces* of our proposed dataset.
This result emphasizes the missing constraints on the train-test split, leading to a high classification accuracy, but disregarding the generalization performance [Valenti2016].
Furthermore, the accuracy of 98.74% relates to the comparatively high accuracy of 89.45%, representing the outcome of ASC approached on the TAU development dataset limited to *Urban Parks* and *Public Squares*, previously presented in Section 4.5.
Additionally, we illustrate precision, recall, and F_1 score in Figure 4.13, along with the confusion matrix, and the normalized confusion matrix in Figure A.5, and Figure 4.14, respectively.

Table 4.10.: Class-wise accuracies, precision, F_1 score, and the number of samples for each class present in the test set, representing the arithmetic mean of 10 independent training runs, of employing ASC on the proposed dataset. For each training run, the samples are randomly distributed throughout the training (33%), validation (33%), and test set (33%).

Acoustic Scenes	Accuracy	Precision	F_1 score	Support
Public Square	99.50%	98.03%	98.75%	119
Urban Green Space	97.98%	99.49%	98.73%	119
Average	98.74%	98.76%	98.74%	
+/-	0.70%	0.68%	0.70%	

Figure 4.13.: Precision, recall, and F_1 score of employing ASC without constraints on our proposed dataset, representing the arithmetic mean over 10 independent training runs. For each training run, the proposed dataset is randomly split in the training (33%), validation (33%), and test set (33%) independently.

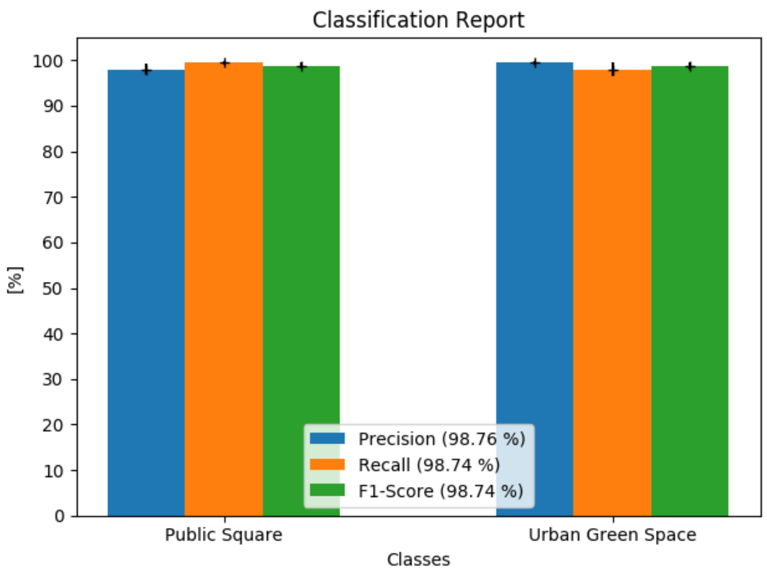
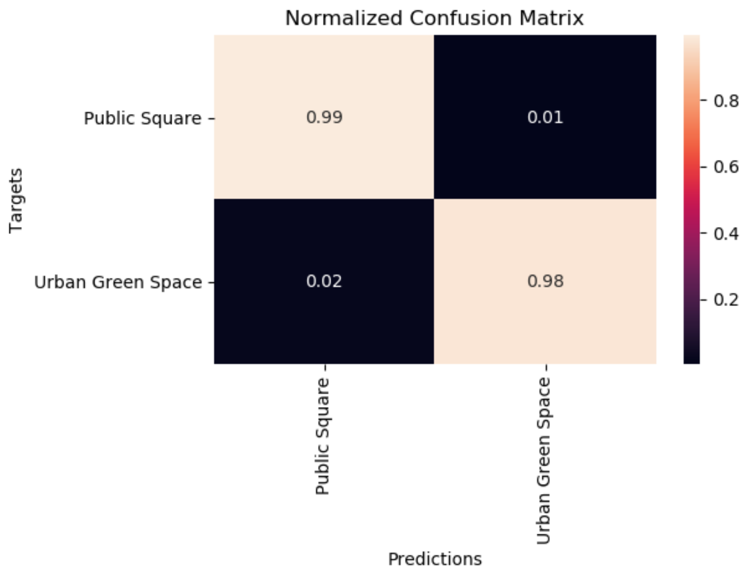


Figure 4.14.: Normalized confusion matrix, representing the arithmetic mean of the class-wise accuracies averaged over 10 independent training runs of employing ASC without constraints on our proposed dataset.



4.8. Proposed Acoustic Scene Classification with Local Constraints

We proceed with ASC on the proposed dataset, restricting samples based on the associated location to either the training, validation, or test set.
Since our proposed dataset provides 2 acoustic scenes, each represented by 4 locations, we approach a 4-2-2 split, i.e., utilizing 4 locations for training, 2 for validation, and 2 for testing. Therefore, we uniformly distribute the samples, w.r.t. the corresponding acoustic scene, throughout the training, validation, and test set.
Additionally, to reduce the degree of uncertainty for the evaluation, we utilize each location for training, validation, and testing, realized by a CV. Therefore, the 4 CV runs, along with the corresponding locations, are represented in Table 4.11.
Based on this 4-2-2 split, the share of samples utilized for the training is represented by 50% of the proposed dataset, as opposed to 33% for the evaluation of ASC without and with temporal constraints, previously presented in Section 4.7, and

subsequently presented in Section 4.9, respectively.
When comparing the results of ASC with local constraints, presented in Table 4.12, to ASC without constraints, presented in Table 4.10, we observe a decrease in the accuracy of 10.59%. More precisely, the accuracy drops from 98.74% to 88.15%. Hence, based on the introduced local constraints, we ensure a reasonable generalization estimate [Valenti2016], resulting in a decrease of the accuracy by 10.59%.
Furthermore, the outcome relates to the comparison between ASC with local constraints and ASC without constraints, employed on the TAU development dataset. Thus, ASC with local constraints employed on the TAU development dataset results in an accuracy of 57.19%, represented in Table 4.4, compared to 76.45% without constraints, represented in Table 4.5. Thus, resulting in a decrease of the accuracy by 19.26% for the evaluation of ASC with local constraints on the TAU development dataset.
Additionally, in Figure 4.15, precision, recall, and F_1 score, averaged over the 4 CV runs, are illustrated, as well as the confusion matrix in Figure A.6, and the normalized confusion matrix in Figure 4.16.
However, since the standard deviation of the accuracy, w.r.t. the CV runs specified in Table 4.11, exceeds 10%, as presented in Table 4.12, we further investigate each run individually.
Therefore, we provide the arithmetic mean of the class-wise accuracies, precision, and F_1 score, averaged over 10 independent training runs, of each CV run, in Table 4.13, Table 4.14, Table 4.15, and Table 4.16.
Thus, the classification performance ranges from 73.00% to 96.33%, and differs at the most by 23.23% between the 2^{nd} and the 1^{st} CV run, presented in Table 4.14, and Table 4.13, respectively.
However, both, the 1^{st} and 4^{th} CV run, reach over 96%, more specifically, 96.33%, and 96.06%, as show in Table 4.13, and Table 4.16. The 3^{rd} run, presented in Table 4.16, results in 87.22%, hence, is ranked between the 1^{st}, and 4^{th} run, respectively, and the 2^{nd} run.
Thus, the practicality of locally constrained ASC is limited by the discrepancy between training and test locations, and hence depends on the knowledge provided by the training data.
Consequently, to investigate the origin of the classification performance for the CV runs, we additionally provide the misclassifications w.r.t. the acoustic scenes, and the underlying locations, for each CV run individually, averaged over 10 independent training runs, in Table 4.17.
Hence, for the 2^{nd} CV run, the location *Am Eisernen Tor* is misclassified as *Urban*

Green Space by 43.89%, and *Herz-Jesu-Kirche* as *Public Square* by 10.11%, resulting in an accuracy of 73%, as presented in Table 4.14.
Since the location *Am Eisernen Tor* exhibits unique background noise, not featured by any other location within our proposed dataset, this high misclassification rate of over 40% might arise from the lack of diversity within the corresponding training data.
The misclassification of the location *Herz-Jesu-Kirche* might occur due to the location's proximity to adjoining streets, and hence is based on the sound of passing cars and trams. Furthermore, *ringing church bells* provide an unknown pattern to the model, thus, might introduce another source of misclassification.
Furthermore, for the 3^{rd} CV run, the location *Lessingstraße 25* is falsely classified as *Public Square* by 20%, limiting the accuracy to 87.22%, as presented in Table 4.15. Therefore, the misclassification rate for the location *Lessingstraße 25* might raise due to the sound of passing cars, occasionally captured within audio samples. However, on the 1^{st} day of recording *Lessingstraße 25*, *construction work* based sound emissions are captured, constituting a more reasonable justification for a proportionally high misclassification rate of 20%.
Consequently, the outcomes emphasize the challenge of classifying locations with an unique soundscape, e.g., *Am Eisernen Tor*, to the corresponding acoustic scene. Furthermore, we highlight the impact of uncommon and location-specific events, e.g., *construction work*, and *ringing church bells*, w.r.t. *Lessingstraße 25*, and *Herz-Jesu-Kirche*, respectively, on the classification performance.
However, to investigate locally constrained ASC in more depth, both, an increased number of temporally independent recordings, and a larger number of diverse locations, constitute prerequisites to increase the variability of locations within acoustic scenes and enable a reasonable generalization.

Table 4.11.: Approached CV for the evaluation of ASC with local constraints on our proposed dataset, utilized to reduce the amount of uncertainty, and hence to avoid unreliable results.

Location	Acoustic Scene	1st	2nd	3rd	4th
Herz-Jesu-Kirche	Urban Green Space	train	test	val	train
Lessingstraße 25	Urban Green Space	train	train	test	val
Naglergasse 35	Urban Green Space	val	train	train	test
Stadtpark	Urban Green Space	test	val	train	train
Am Eisernen Tor	Public Square	train	test	val	train
Hauptplatz	Public Square	train	train	test	val
Jakominiplatz	Public Square	val	train	train	test
Kaiser-Josef-Platz	Public Square	test	val	train	train

Table 4.12.: Arithmetic mean of the class-wise accuracies, precision, and F_1 score, averaged over 10 independent training runs of each run w.r.t. the CV, presented in Figure 4.11, along with the number of samples provided for each acoustic scene within in the test set. Hence, providing the results of employing ASC with local constraints, utilizing 50% of our proposed dataset for training, and 25% for validation, and testing, respectively.

Acoustic Scenes	Accuracy	Precision	F_1 score	Support
Public Square	85.50%	90.53%	86.77%	90
Urban Green Space	90.81%	88.56%	88.91%	90
Average	88.15%	89.54%	87.84%	
+/-	10.48%	9.15%	10.99%	

Figure 4.15.: Illustration of the arithmetic mean w.r.t. the approached CV presented in Figure 4.11 with 10 independent training runs for each CV run, representing the averaged precision, recall, and F_1 score, based on the evaluation of ASC with local constraints on our proposed dataset.

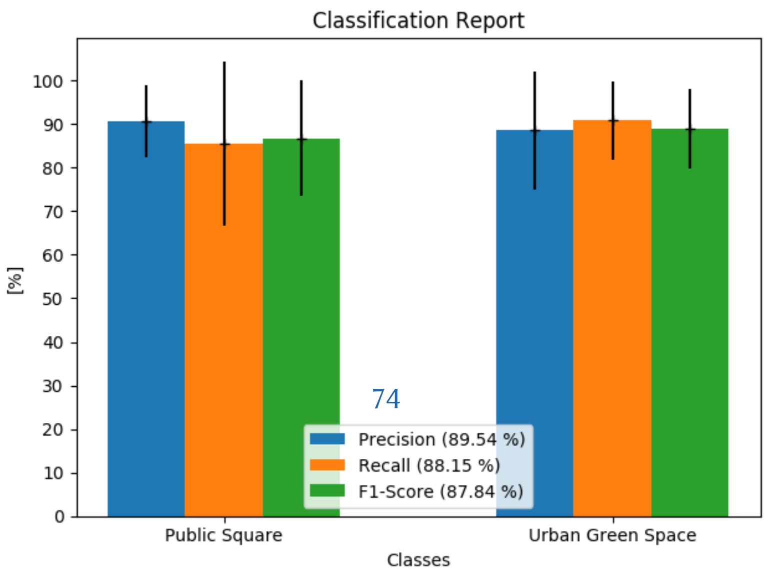


Figure 4.16.: Normalized confusion matrix, providing the arithmetic mean of the class-wise accuracies w.r.t. the CV with 10 independent training runs for each CV run, utilized for the evaluation of ASC with local constraints on our proposed dataset.

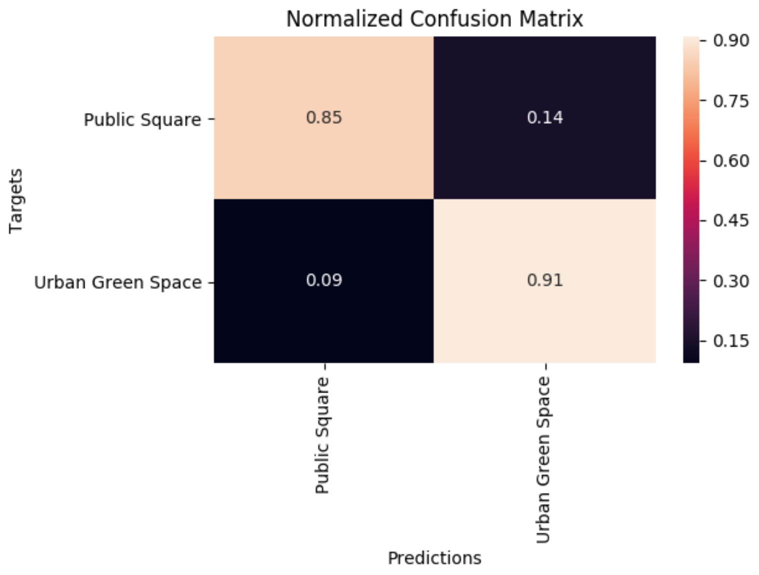


Table 4.13.: Arithmetic mean of the class-wise accuracies, precision, and F_1 score, averaged over 10 independent training runs, of the 1st run w.r.t. the CV, of locally constrained ASC on our proposed dataset.

Acoustic Scenes	Accuracy	Precision	F_1 score	Support
Public Square	92.67%	100.00%	96.08%	90
Urban Green Space	100.00%	93.48%	96.55%	90
Average	96.33%	96.74%	96.31%	
+/-	3.18%	2.66%	3.21%	

Table 4.14.: Arithmetic mean of the class-wise accuracies, precision, and F_1 score, based on the 2^{nd} run of the CV w.r.t. locally constrained ASC on our proposed dataset, averaged over 10 independent training runs.

Acoustic Scenes	Accuracy	Precision	F_1 score	Support
Public Square	56.11%	85.47%	66.85%	90
Urban Green Space	89.89%	67.69%	76.97%	90
Average	73.00%	76.58%	71.91%	
+/-	5.31%	4.59%	6.11%	

Table 4.15.: Arithmetic mean of the class-wise accuracies, precision, F_1 score, of the 3^{rd} run w.r.t. the CV, hence, averaged over 10 independent trainings runs, of ASC with local constraints on our proposed dataset.

Acoustic Scenes	Accuracy	Precision	F_1 score	Support
Public Square	94.44%	82.91%	88.01%	90
Urban Green Space	80.00%	94.30%	86.18%	90
Average	87.22%	88.60%	87.09%	
+/-	6.35%	6.33%	6.42%	

4.9. Proposed Acoustic Scene Classification with Temporal Constraints

Next, we approach ASC on the proposed dataset, restricting samples recorded on the same date to either the training, validation, or test set. Thus, we propose temporal constraints for the application of ASC.

Since our proposed dataset is composed of samples recorded on 3 consecutive days, we obtain 3 equally sized folds. Hence, to reduce the degree of uncertainty for the evaluation, we train and test the permutations w.r.t. the 3 folds, instead of

Table 4.16.: Arithmetic mean, averaged over 10 independent training runs, of the class-wise accuracies, precision, and F_1 score, w.r.t. the 4^{th} run of the CV, of ASC with local constraints on our proposed dataset.

Acoustic Scenes	Accuracy	Precision	F_1 score	Support
Public Square	98.78%	93.75%	96.16%	90
Urban Green Space	93.33%	98.77%	95.94%	90
Average	96.06%	96.26%	96.05%	
+/-	1.04%	0.99%	1.04%	

employing 10 independent training runs on the same train-test split.
Thus, the results represent the arithmetic mean of the 6 training runs w.r.t. the permutations of the 3 folds, utilizing each fold for training, validation, and testing. In addition, this approach avoids biased results by evaluating the performance solely on the best, or a random, train-test permutation.
Thus, in Table 4.18, we present the arithmetic mean of the class-wise accuracies w.r.t. the 6 training runs. Furthermore, precision, F_1 score, and the number of samples for each acoustic scene of the test set are presented in Table 4.18.
As the average accuracy reaches 95.14%, we observe a decrease of 3.6%, compared to 98.74% when employing ASC without constraints on our proposed dataset, presented in Table 4.10.
This decrease in the accuracy of 3.6% arises from the corrected generalization estimate [Valenti2016], and hence is based on the introduced temporal constraints.
Furthermore, utilizing on temporal constraints, when employing ASC on the proposed dataset, results in a comparatively high accuracy of 95.14%, represented in Table 4.18, compared to 88.06% when employing ASC with local constraints, represented in Table 4.12.
This decrease in the accuracy of $7,08\%$, when employing ASC with local constraints, might arise from the relatively low number of 4 locations per acoustic scene provided by our proposed dataset, and hence is based on insufficient knowledge to generalize locations to acoustic scenes.
Additionally, precision, recall, and F_1 score, are illustrated in Figure 4.17.
Finally, the confusion matrix, as well as the normalized confusion matrix, are represented in Figure A.7, and Figure 4.18, respectively.

Table 4.17.: In-depth analysis of the misclassified acoustic scenes for each run of the CV w.r.t. locally constrained ASC on our proposed dataset, and hence presenting the arithmetic mean and standard deviation of the number of misclassified samples, along with the underlying location, for each acoustic scene and CV run, averaged over 10 independent training runs.

CV run	Target	Location	avg. #	+/-	avg. %	+/-
1	PS	Kaiser-Josef-Platz	7	6	7.33%	6.37%
2	PS	Am Eisernen Tor	40	11	43.89%	11.72%
	UGS	Herz-Jesu-Kirche	9	6	10.11%	6.41%
3	PS	Hauptplatz	5	8	5.56%	9.08%
	UGS	Lessingstraße 25	18	8	20.00%	8.65%
4	PS	Jakominiplatz	1	2	1.22%	1.82%
	UGS	Naglergasse 35	6	2	6.67%	2.43%

Figure 4.17.: Illustration of the arithmetic mean of precision, recall, and F_1 score, averaged over 6 training runs, representing the permutations of the 3 folds established for the evaluation of ASC with temporal constraints on our proposed dataset. Hence, restricting samples by the date of recording to either the training, validation, or test set.

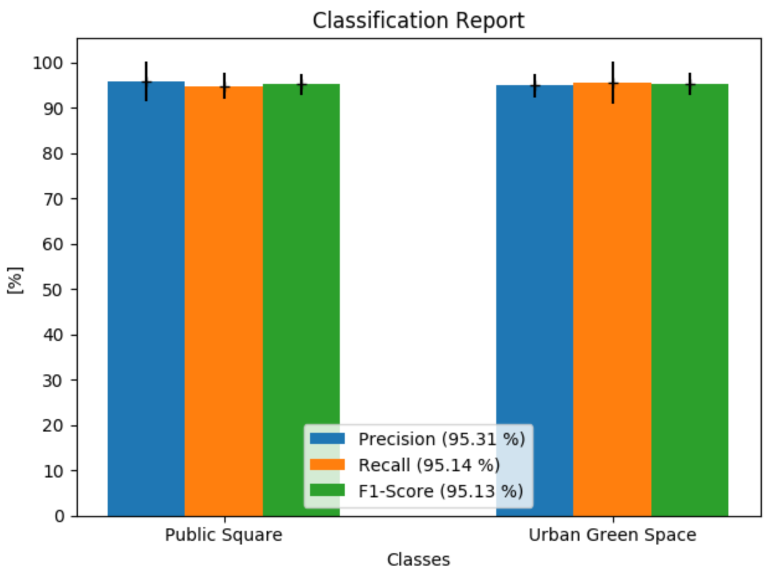
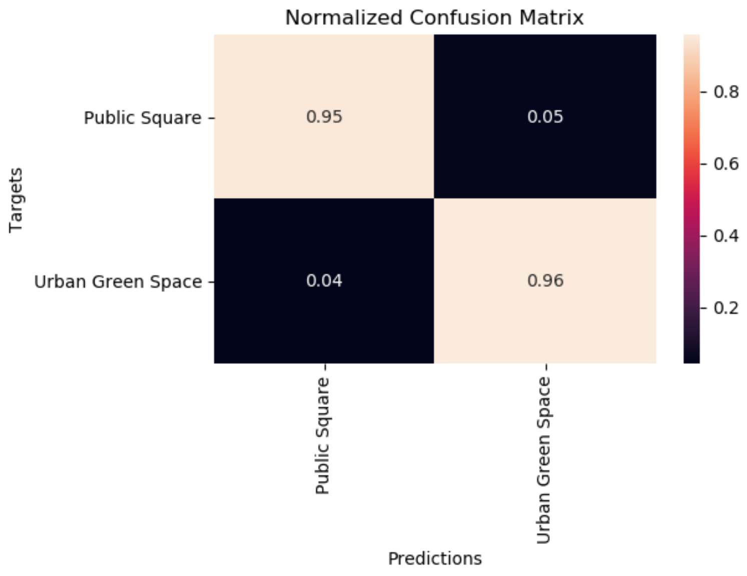


Table 4.18.: Arithmetic mean of the class-wise accuracies, precision, and F_1 score, averaged over 6 independent training runs of the employed ASC with temporal constraints on our proposed dataset. Hence, the 6 training runs constitute the permutations of the 3 equally sized folds, established by splitting the samples of the proposed dataset w.r.t. to the date of recording.

Acoustic Scenes	Accuracy	Precision	F_1 score	Support
Public Square	94.72%	95.75%	95.15%	120
Urban Green Space	95.56%	94.86%	95.12%	120
Average	95.14%	95.31%	95.13%	
+/-	2.37%	2.31%	2.37%	

Figure 4.18.: Normalized confusion matrix, i.e., the class wise accuracies, of employing ASC with temporal constraints on our proposed dataset. Hence, representing the arithmetic mean of 6 independent training runs, constituting the permutations w.r.t. the 3 folds, each providing samples recorded on a different date.



4.9.1. Humans versus Machines

To evaluate the human ability of classifying audio samples into acoustic scenes, and hence to provide a benchmark for the performance of artificial ASC, we conduct a human based ASC evaluation on a subset of our dataset.
Therefore, 3 test persons attempt to classify 10 s audio samples into the acoustic scenes *Public Square* and *Urban Green Space*. Hence, to adapt the volume level to the human auditory, the audio samples are amplified by 300%.
To provide each test person with the same knowledge base, 1 audio sample of each location, hence, 4 audio samples of *Public Squares*, and *Urban Green Spaces*, respectively, are introduced prior the classification. Additionally, photographies of the locations are presented and characteristics of the locations are described, e.g., the fountain at the location *Am Eisernen Tor*. After the *training*, 16 audio samples, 2 of each location, are classified by each test person independently. Therefore, the corresponding audio sample is provided once to each test person, and the classification takes place subsequently. Additionally, to increase the complexity, the samples of the training and the test set are temporally constrained, hence, are recorded on a different date.
Thus, in Table 4.19, we present the outcomes of the human based ASC, yielding an average accuracy of 95.83%.
However, 2 of the 3 test persons are resident in the area where the recordings are conducted—1 of those 2 participants has been involved in the recording process as well—both achieving a perfect classification. The 3^{rd} test person, not resident in the recorded area, misclassified the acoustic scene *Urban Green Space*, more specifically, the location *Herz-Jesu-Kirche*, twice as *Public Square*.
This misclassification might be deduced from the noise of passing vehicles in the test samples, opposed to hardly humanly audible noise in the training sample. Furthermore, since the test persons have knowledge of the location's distribution within the test set, i.e., 2 samples per location, empirical decision making might be an issue in terms of unnecessary misclassifications.
Summarized, humans slightly outperform artificial ASC within a temporally constrained setup, resulting in an accuracy of 95.83%, compared to 95.14% when artificially classified, as presented in Table 4.18. Furthermore, if the worst performing test person is excluded from the evaluation, humans achieve a perfect classification performance, even outperforming unconstrained artificial ASC with an accuracy of 98.74%, as presented in Table 4.10. However, the number of parti-

cipants, and the number of evaluated samples, might be insufficient to draw final conclusions.

Table 4.19.: Arithmetic mean of the class-wise accuracies, precision, and F_1 score, along with the number of evaluated samples, averaged over the 3 participants of the *Humans versus Machines* survey, hence, estimating the human ability of classifying audio samples of our proposed dataset into acoustic scenes.

Acoustic Scenes	Accuracy	Precision	F_1 score	Support
Public Square	100.00%	93.33%	96.30%	8
Urban Green Space	91.67%	100.00%	95.24%	8
Average	95.83%	96.67%	95.77%	
+/-	5.89%	4.71%	5.99%	

4.10. Proposed Acoustic Location Classification

In the following, we introduce ALC on the proposed dataset, starting off the evaluation without constraints on the train-test split.
To provide comparable results w.r.t. ALC with temporal constraints, subsequently presented in Section 4.11, we split the dataset into 3 equally sized folds—since our proposed dataset is assembled on 3 consecutive working days, implementing temporal constraints for ALC results in the stated uniform distribution.
Thus, we utilize 66% of the dataset for training, and the remaining 33% for testing. Moreover, 50% of the training data is utilized for the validation set to enable ES, and hence is employed to identify the best trained model, i.e., the model trained with the number of epochs resulting in the lowest validation loss. Furthermore, the validation set is utilized to stop the training after 20 epochs of non-improvement. For the evaluation of ALC on the proposed dataset, we conduct 10 independent trainings runs, randomly splitting the dataset for each run independently, while uniformly distributing the samples w.r.t. the associated locations throughout the folds.
As presented in Table 4.20, which provides the class-wise accuracies, along with precision, F_1 score, and the number of samples per location of the test set, we achieve

a class-wise accuracy, averaged over 10 independent training runs, of 85.17%. Besides, the number of samples provided for each class, represented by the rounded arithmetic mean over the 10 independent training runs, might deviate by one sample due to the employed random split implementation.
We attribute this comparatively high accuracy of 85.17% to the missing constraints on the train-test split, disregarding the actual generalization performance [Valenti2016], and hence providing an overfitted model [Mesaros2018a] w.r.t. the recordings of the proposed dataset.
Furthermore, the outcome of 85.17% relates to the accuracy of employing ALC on the TAU development dataset, limited to locations associated with the acoustic scene *Urban Park*, resulting in an accuracy of 84.27%, as presented in Table 4.9.
When investigating the normalized confusion matrix, illustrated in Figure 4.20, we observe notable misclassifications: *Hauptplatz* is misclassified as *Kaiser-Josef-Platz* by 22%, which might be explained with the recording of the 3^{rd} day. More precisely, based on a slightly different position and a higher population of people, the humanly produced noise might excel the background of the fountain located at *Hauptplatz*. Furthermore, *Jakominiplatz* is misclassified as *Kaiser-Josef-Platz* by 11%, and vice versa by 12%. This misclassification seems reasonable due to the common properties of both locations, i.e., emitting noise based on the population of people as well as on passing vehicles, i.e., trams and buses.
Additionally, we illustrate precision, recall, and F_1 score in Figure 4.19, along with the confusion matrix in Figure A.8.

Table 4.20.: Arithmetic mean of the class-wise accuracies, precision, and F_1 score, averaged over 10 independent training runs, along with the number of samples provided in the test set, representing the results of employing ALC without constraints on our proposed dataset. To ensure a reliable comparison for the evaluation of ALC with temporal constraints w.r.t. to the number of samples utilized for training, we split the proposed dataset into the training, validation, and test set, each representing 33%.

Locations	Accuracy	Precision	F_1 score	Support
Am Eisernen Tor	97.99%	94.88%	96.38%	29
Hauptplatz	73.99%	86.12%	79.48%	29
Herz-Jesu-Kirche	74.30%	83.45%	78.01%	29
Jakominiplatz	88.78%	88.38%	88.41%	29
Kaiser-Josef-Platz	78.17%	67.84%	72.45%	29
Lessingstraße 25	83.55%	85.84%	84.06%	29
Naglergasse 35	91.32%	88.68%	89.72%	29
Stadtpark	93.26%	92.40%	92.66%	29
Average	85.17%	85.95%	85.15%	
+/-	1.70%	1.63%	1.69%	

Figure 4.19.: Arithmetic mean of precision, recall, and F_1 score, representing the results of employing ALC without constraints on our proposed dataset, hence averaged over 10 independent training runs. Training, validation, and test set constitute 33% of the proposed dataset respectively to provide a reasonable benchmark for ALC with temporal constraints w.r.t. to the number of utilized training samples. Furthermore, training, validation, and test set are randomly assembled for each training run independently.

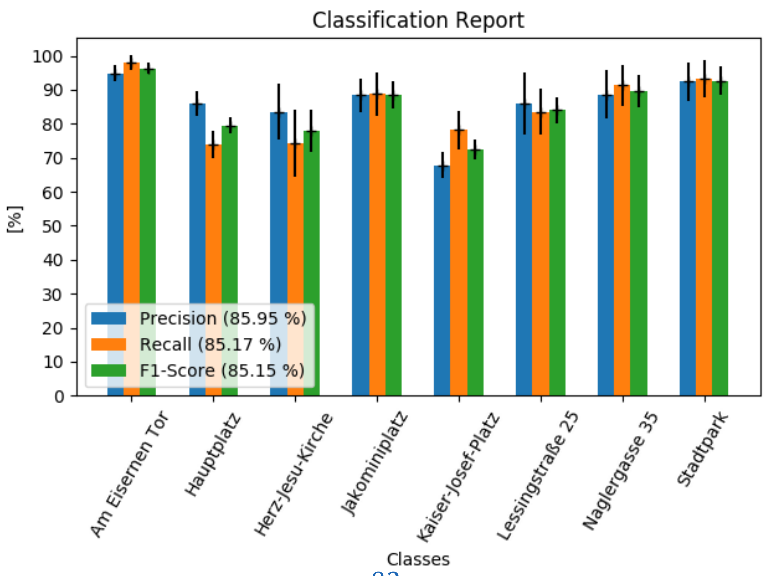
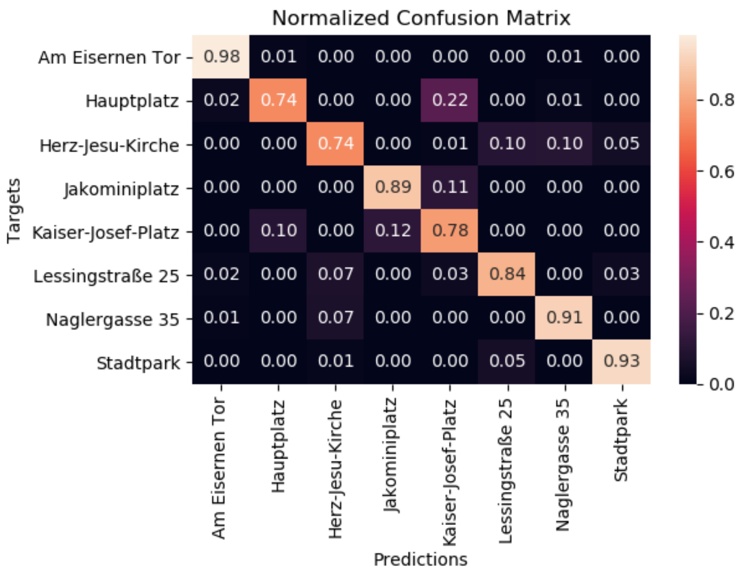


Figure 4.20.: Normalized confusion matrix of employing unconstrained ALC on our proposed dataset, hence, representing the arithmetic mean of the class-wise accuracies over 10 independent training runs, while splitting the dataset into training (33%), validation (33%), and test set (33%) for reach run independently.



4.11. Proposed Acoustic Location Classification with Temporal Constraints

Finally, we present the evaluation of ALC on the proposed dataset, introducing temporal constraints by splitting the training and the test set to constitute different dates. This temporal constraints enable a reasonable and empirical investigation w.r.t. the practicality and the limitations of classifying audio samples to the location of origin.

Thus, we split the samples of the proposed dataset by the date of recording and receive 3 equally sized folds, each representing 33% of the dataset.

To avoid biased results by justifying the evaluation solely on the best, or a random fold, the permutations of the 3 folds are utilized for training, validation, and testing, as specified in Table 4.21. Hence, we provide the arithmetic mean of the 6 CV runs as result.

Thus, we train the CNN on the particular training set, interrupt the training after the validation loss stops improving for at least 20 epochs, select the most generalized model, i.e., the model trained with the number of epochs resulting in the lowest error w.r.t. the validation set, and evaluate the selected model on the test set.
Therefore, we present the outcomes, i.e., the class-wise accuracies, precision, and F_1 score, averaged over the 6 CV runs, and the number of samples for each location of the test set, in Table 4.22. Thus, we achieve an accuracy of 63.82%, constituting a decrease of 21.35% compared to 85.17% of employing unconstrained ALC on the proposed dataset, as presented in Table 4.20.
This decrease of 21.35% indicates, based on the temporal constraints, a rectification of the stated classification performance w.r.t. generalization [Valenti2016]. Thus, despite the worse classification accuracy when implementing temporal constraints, the outcomes represent the possibilities and limitations of ALC in a more reliable way.
Furthermore, the normalized confusion matrix, representing ALC with temporal constraints, illustrated in Figure 4.22, correlates with the corresponding confusion matrix of unconstrained ALC, illustrated in Figure 4.20, apart from a throughout worse classification performance. For example, *Hauptplatz* is misclassified as *Kaiser-Josef-Platz* by 46%, compared to a misclassification of 22%, when employing ALC without constraints on the proposed dataset.
This indicates that introducing temporal constraints dampens the overall classification accuracy, while preserving susceptible classes for misclassification.
Additionally, precision, recall, and F_1 score are illustrated in Figure 4.21, along with the confusion matrix in Figure A.9.
To investigate the outcome of ALC with temporal constraints on our proposed dataset in more depth, we present the arithmetic mean of the class-wise accuracies, precision, and F_1 score, averaged over 10 independent training runs w.r.t. the 1^{st} CV run, i.e., utilizing the 1^{st} day for training, the 2^{nd} day for validation, and the 3^{rd} day for testing, in Table 4.23. Additionally, we illustrate the classification report, the confusion matrix, and the normalized confusion matrix, of the 1^{st} CV run, in Figure 4.23, Figure A.10, and Figure 4.24, respectively.
Thus, as presented in Figure 4.24, *Hauptplatz* is misclassified by 100%, falsely classified as *Kaiser-Josef-Platz* by 91%.
This high misclassification of 100% for *Hauptplatz* might be explained by the position change for the recording on the 3^{rd} day as well as on the higher population of people compared to the 1^{st} and 2^{nd} day. Thus, the samples recorded on the 3^{rd} day might miss information on the fountain, included in the recordings of the first 2

days.
This behavior emphasizes the limitations of ALC, especially, when employed on a dataset with a limited number of samples. Thus, the model might not be able to learn a generalized representation of locations based on a single day of training data. However, learning a generalized representation of locations over a longer period of time might become infeasible due to an arising intraclass variability [Schmidhofer2018].
Summarized, an extended dataset, providing daily audio recordings over a longer time frame, i.e., over several weeks or months, might be necessary to investigate the limitations of ALC with temporal constraints to a sufficient degree.

Table 4.21.: CV runs, i.e., permutations, of the approached 3-fold CV with temporal constraints on training, validation, and test set, for the evaluation of ALC on our proposed dataset.

CV folds vs. runs	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	6^{th}
1^{st} day	train	train	val	test	test	val
2^{nd} day	val	test	test	val	train	train
3^{rd} day	test	val	train	train	val	test

Table 4.22.: Arithmetic mean of the class-wise accuracies, precision, and F_1 score, along with the number of samples for each class provided by the test set, averaged over the 6 CV runs w.r.t. to the permutations of the introduced 3-fold CV for the evaluation of ALC with temporal constraints on our proposed dataset.

Locations	Accuracy	Precision	F_1 score	Support
Am Eisernen Tor	91.11%	91.49%	90.94%	30
Hauptplatz	38.33%	34.74%	32.51%	30
Herz-Jesu-Kirche	56.67%	61.13%	57.62%	30
Jakominiplatz	72.78%	67.64%	69.61%	30
Kaiser-Josef-Platz	25.00%	26.02%	22.29%	30
Lessingstraße 25	46.67%	58.43%	48.63%	30
Naglergasse 35	86.11%	81.84%	83.51%	30
Stadtpark	93.89%	78.45%	85.09%	30
Average	63.82%	62.47%	61.27%	
+/-	4.49%	5.30%	4.29%	

Figure 4.21.: Illustration of the arithmetic mean of precision, recall, and F_1 score, averaged over the 6 CV runs w.r.t. the permutations of the temporally constrained train-test split for ALC on our proposed dataset.

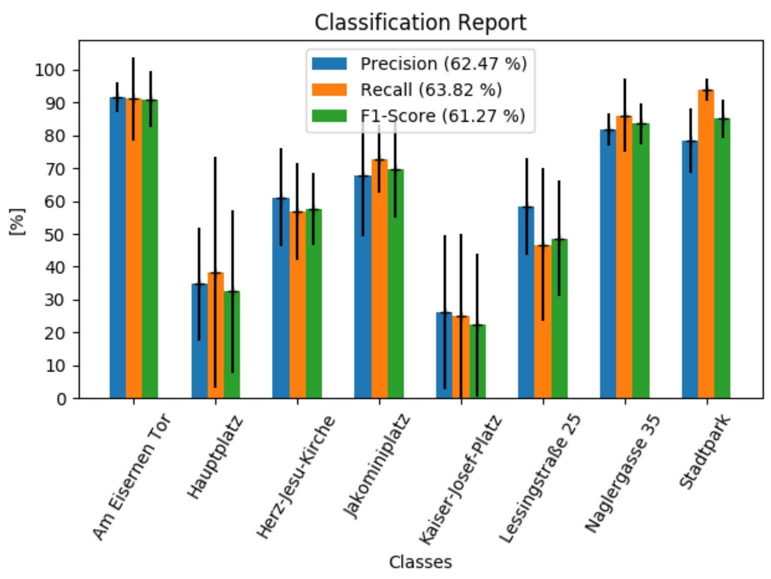


Figure 4.22.: Normalized confusion matrix, representing the arithmetic mean of the class-wise accuracies w.r.t. the permutations, thus, averaged over the 6 CV runs, and hence representing the results for the employed ALC with temporal constraints on our proposed dataset.

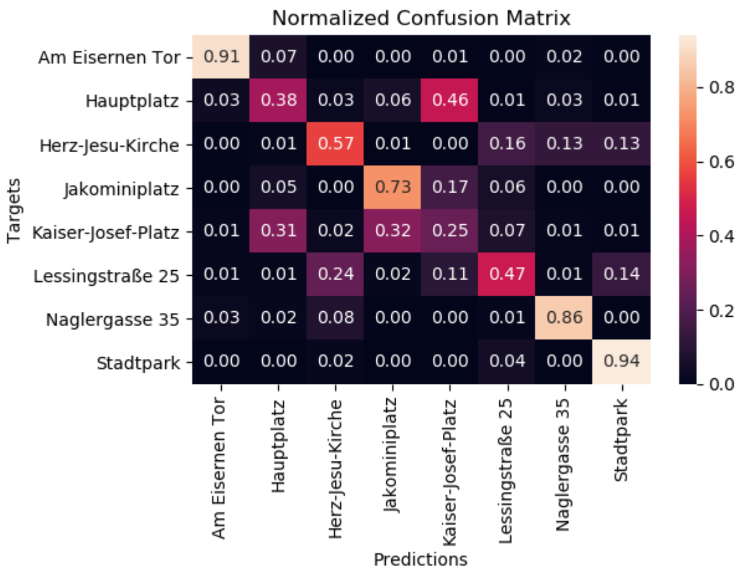


Table 4.23.: Arithmetic mean of the class-wise accuracies, precision, and F_1 score, as wells as the number of samples provided for each class by the test set, averaged over 10 independent training runs of the 1^{st} CV run for ALC on our proposed dataset. Thus, utilizing the 1^{st} day, i.e., the samples recorded on the 1^{st} day, for training, the 2^{nd} day for validation, and the 3^{rd} day for testing.

Locations	Accuracy	Precision	F_1 score	Support
Am Eisernen Tor	99.00%	91.99%	95.27%	30
Hauptplatz	0.00%	0.00%	0.00%	30
Herz-Jesu-Kirche	54.67%	59.64%	56.89%	30
Jakominiplatz	81.00%	98.58%	88.27%	30
Kaiser-Josef-Platz	84.33%	43.25%	56.83%	30
Lessingstraße 25	63.67%	65.58%	63.64%	30
Naglergasse 35	71.00%	80.48%	74.65%	30
Stadtpark	95.00%	73.09%	82.41%	30
Average	68.58%	64.08%	64.75%	
+/-	3.04%	2.00%	2.84%	

Figure 4.23.: Arithmetic mean of precision, recall, and F_1 score, averaged over 10 independent training runs of the 1^{st} CV run w.r.t. the 3-fold CV for ALC with temporal constraints on the proposed dataset.

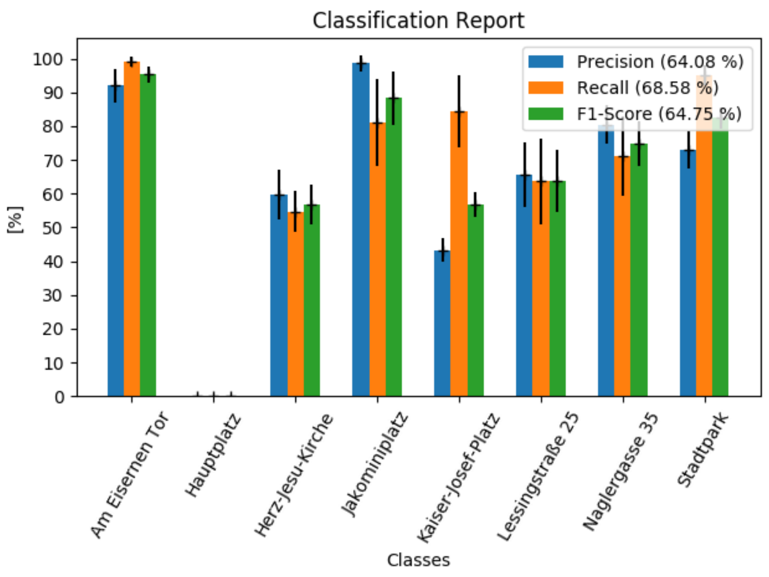
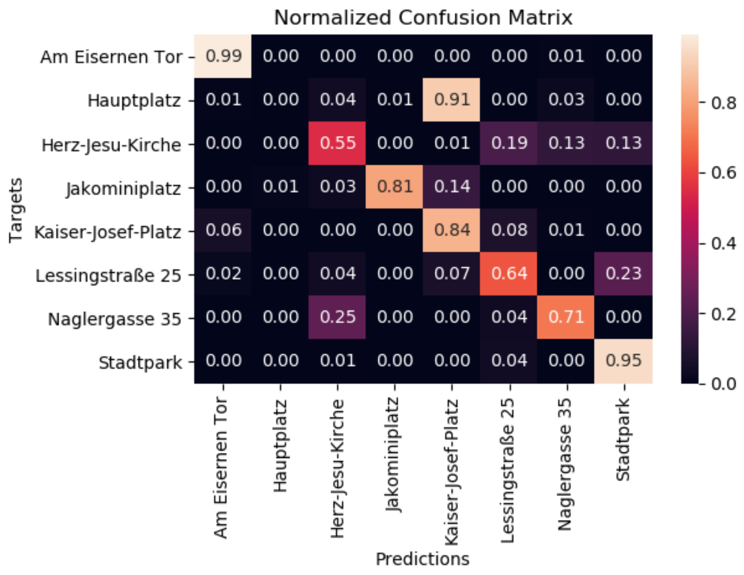


Figure 4.24.: Normalized confusion matrix, hence, the arithmetic mean of the class-wise accuracies, averaged over 10 independent training runs, and thus representing the outcomes of the 1^{st} CV run for the evaluation of ALC with temporal constraints on our proposed dataset. Therefore, samples recorded on the 1^{st} day are utilized for training, samples on the 2^{nd} day for validation, and samples on the 3^{rd} day for testing.



4.11.1. Number of training samples versus classification performance

In addition, to determine the required recording time, and the number of samples, respectively, for temporally constrained ALC, we investigate the correlation between the number of training samples and the classification performance.
Therefore, we split our proposed dataset into 3 uniformly distributed folds, restricting samples by the date of recording to one of the 3 folds. Additionally, to avoid a biased estimation w.r.t. the convergence behavior, we evaluate the permutations of the 3 folds, as specified in Table 4.21.
Thus, for each CV run, we start off with 1 training sample per location, and increment the number of samples per location by 1, until we reach 30 training samples per location, and hence 240 training samples in total, i.e., the entire training set. Therefore, we train and evaluate 30 models for each of the 6 CV runs, totalling 180 models. However, the corresponding validation and test set remain unchanged.
Thus, in Figure 4.25, we illustrate the impact of the number of training samples on the classification performance for each CV run and training sample count, along with the arithmetic mean, averaged over the 6 CV runs, and the corresponding rolling average, more specifically, rolling arithmetic mean, with window size 5.
In addition, to estimate the number of required training samples, the horizontal line indicates the last rolling average of the accuracy, i.e., the rolling average of the arithmetic mean averaged over the accuracies of the models trained on the entire training set, and the vertical line approximates the end of convergence.
Thus, we estimate the number of training samples, required to provide a sophisticated model w.r.t. temporally constrained ALC on our proposed dataset, with 160 samples, i.e., 20 samples per location. Hence, we propose a minimum recording time of 200 s per location w.r.t. the training set.
However, this estimated number of required training samples arises from the evaluation of temporally constrained ALC based on our proposed dataset, and hence might not hold for related datasets and similar problem statements.
Furthermore, the reliability of the estimation is restricted by the limited number of 3 recording days, and 8 locations, respectively, provided by our proposed dataset. Hence, extending the size of the dataset constitutes a prerequisite for a more sophisticated evaluation.
In addition, estimating the end of convergence, and hence the number of required training samples, on a visual basis constitutes a highly subjective task and might

result in a biased outcome.

4.12. Summary

Finally, in Table 4.24, we present the main results of our experiments for the investigation of ASC and ALC w.r.t. local and temporal constraints on the TAU development dataset and our proposed dataset. However, to provide the necessary context, adjoining experiments are included in the list as well. Hence, the most import results are explicitly highlighted.

The first noteworthy outcome concerns the transferred ASC model, trained on the acoustic scenes *Urban Park* and *Public Square* of the TAU development dataset, and evaluated on our proposed dataset, presented in Section 4.5.1. Hence, we achieve an accuracy of 93% for the evaluation of the transferred model on our proposed dataset, compared to 88.15% for locally constrained ASC, and 95.14% for temporally constrained ASC, trained and evaluated on our proposed dataset, although transferring the model implies local and temporal constraints by nature. Hence, we highlight the capability of transferring models between related datasets.

Next, we refer to unconstrained ASC on our proposed dataset, presented in Section 4.7, resulting in an almost perfect classification of 98.74%. To verify this high result, and to investigate the human ability to perform ASC in general, we conduct a study with 3 test persons on a subset of our proposed dataset, resulting in an accuracy of 95.83%, as presented in Section 4.9.1. However, although the training and test samples provided to the test persons are temporally constrained, opposed to the artificial approach, 2 out of the 3 test persons achieve a perfect classification, hence, empirically verify the correctness of the results of unconstrained ASC on our proposed dataset. Furthermore, by reaching 95.83%, humans even outperform the artificial counterpart, i.e., temporally constrained ASC on our proposed dataset with an accuracy of 95.14%, as presented in Section 4.9.

Moreover, we emphasize the outcome of locally constrained ASC based on our proposed dataset, presented in Section 4.8. Therefore, we approach a $4-2-2$ split w.r.t. the locations, and, to ensure reliable outcomes, perform a CV to evaluate the predictiveness of each location in particular. However, the standard deviation w.r.t. the CV runs exceeds 10%, more specifically, reaches 10.48%. Hence, each CV run is investigated independently, and the origin of misclassification is discussed. Therefore, we highlight the negative impact of uncommon events captured within

recordings as well as the issue of unique soundscapes, i.e., locations with a soundscape incomparable to a location within the training set.
Next, we point to the outcome of ALC on our proposed dataset, more precisely, the outcome of temporally constrained ALC, as presented in Section 4.11. Therefore, we split the dataset based on date of recording into 3 uniformly distributed folds. Furthermore, we perform a CV w.r.t. the fold-permutations, resulting in an accuracy of 63.82%, and hence in a decrease of 21.35%, compared to unconstrained ALC on our proposed dataset with an accuracy of 85.17%, as presented in Section 4.10. However, temporally constrained ALC yields a standard deviation of 4.49% w.r.t. the 6 CV runs. Therefore, we investigate the 1^{st} CV run in more depth, and hence showcase the impact of temporal gaps between training and test data.
Finally, we emphasize the impact of the training set size on the classification performance w.r.t. temporally constrained ALC on our proposed dataset, presented in Section 4.11.1. Therefore, we split the dataset based on the date of recording into 3 equally sized folds, and perform a CV w.r.t. the fold-permutations. Furthermore, for each CV run, we evaluate multiple models based on a different training set size, more precisely, we start with 1 training sample per location, and increment the number of samples per location by 1 till 30 samples per location, totalling 240 training samples. Hence, we investigate the correlation between the number of training samples and the classification performance. Consequently, we approximate the minimum training set size, required for a sophisticated model w.r.t. temporally constrained ALC on our proposed dataset, with 20 10 s audio samples, i.e., 200 s audio data, per location.

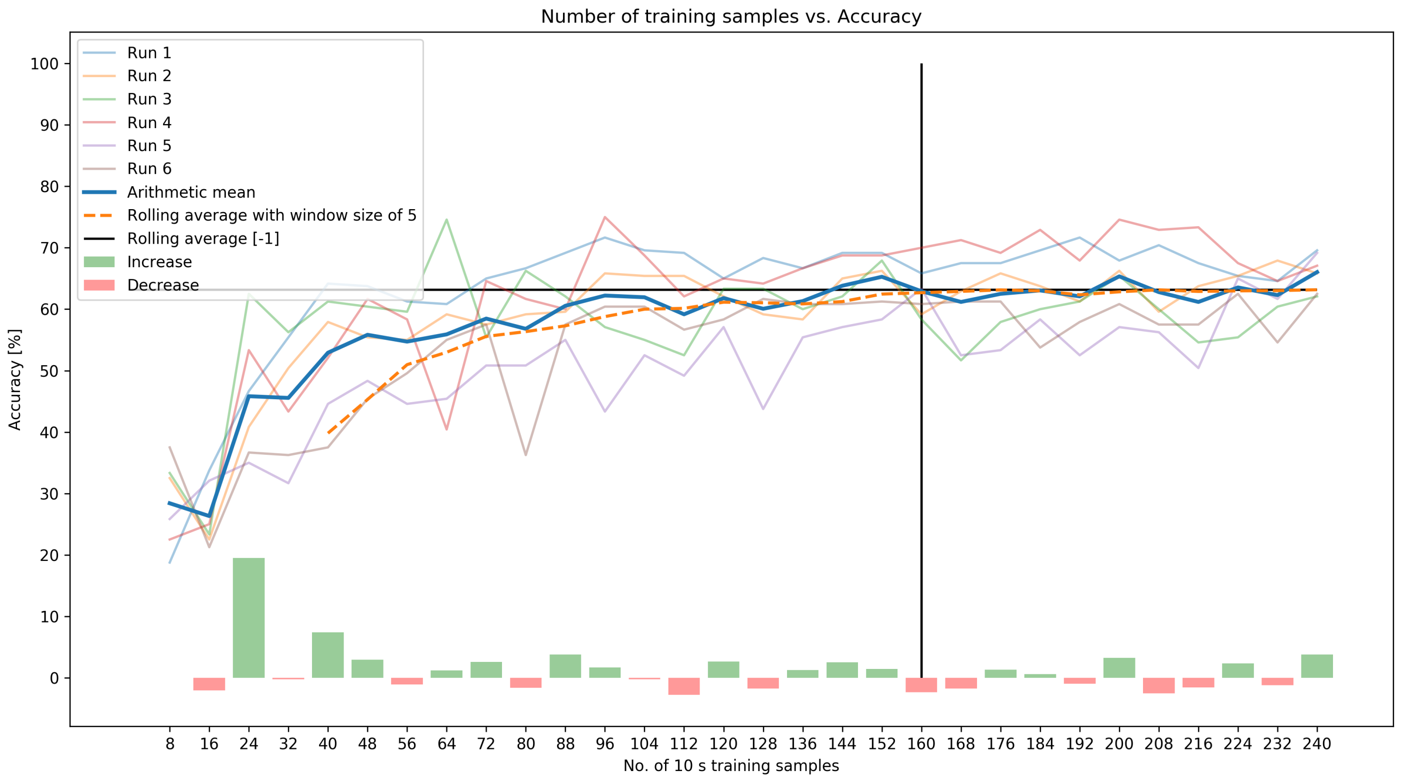


Figure 4.25.: Class-wise accuracies for the 6 CV runs w.r.t. ALC with temporal constraints based on our proposed dataset, along with the arithmetic mean and the corresponding rolling average with window size of 5, for different numbers of training samples.

Table 4.24.: Results of evaluating ASC and ALC w.r.t. local and temporal constraints based on the TAU development dataset and our proposed dataset. Unexpected, extreme, and noteworthy results are highlighted.

Summary of Results			
Reference	Objective	Accuracy	+/-
4.3	ASC on TAU Dev DS with Local Constrains	57, 19%	2, 30%
4.4	ASC on TAU Dev DS without Constraints	76, 45%	1, 78%
4.5	ASC on TAU Dev DS on Urban Parks & Public Squares with Local Constraints	89, 45%	0, 51%
4.5.1	ASC trained on TAU Dev DS on Urban Parks & Public Squares with Local Constraints & evaluated on Proposed DS	**93,00%**	1, 51%
4.6	ALC on TAU Dev DS on Urban Parks without Constraints	84, 27%	1, 75%
4.7	ASC on Proposed DS without Constraints	**98,74%**	0, 70%
4.8	ASC on Proposed DS with Local Constraints	**88,15%**	**10,48%**
4.9	ASC on Proposed DS with Temporal Constraints	**95,14%**	2, 37%
4.9.1	ASC on Proposed DS with Temporal Constraints by Humans	**95,83%**	5, 89%
4.10	ALC on Proposed DS without Constraints	**85,17%**	1, 70%
4.11	ALC on Proposed DS with Temporal Constraints	**63,82%**	**4,49%**
4.11.1	ALC on Proposed DS with Temporal Constraints for varying numbers of training samples	**. . . %**	. . . %

5. Conclusions

To establish a dataset of audio samples with the necessary metadata and properties to investigate the artificial classification of audio samples to locations, we implemented a client-server solution based on Android smartphones as recording devices. Utilizing a dedicated mobile application, we collected audio samples at 8 different locations in Graz (AUT), represented by the acoustic scenes *Urban Green Space* and *Public Square*, on 3 consecutive working days. Thus, we proposed a dataset, providing 5 minutes of audio data for each day, and location, respectively, totalling 720 10 s monophonic audio samples with a sample rate of 48 kHz and a bit depth of 16 bit. Furthermore, we implemented a 2-layer deep CNN to classify 10 s audio samples based on 40-band log-mel spectrograms with a window size of 40 ms, an overlap of 20 ms, and hamming windows.
With the goal of investigating the practicality and the limitations of artificially classifying audio samples to the location of origin, denoted as ALC, we conducted several experiments on our proposed dataset, and, to provide a benchmark, on the TAU development dataset, provided for the DCASE 2019 ASC challenge[1].
Thus, we evaluated the TAU development dataset by approaching ASC, the task of classifying audio samples to an abstraction of the represented location, to investigate the impact of locally constraining training and test sets, i.e., restricting audio samples associated with one particular location to either the training or the test set.
As demonstrated by our results, removing the local constraints directly increases the classification accuracy. However, removing the local constraints also tampers the generalization estimate [Valenti2016] and enables the model to overfit [Mesaros2018a].
Furthermore, we investigated ALC on the TAU development dataset, achieving an accuracy of 84.27% for the classification of locations within the acoustic scene *Urban Park*. Since restricting audio samples based on the location of origin to either

[1]http://dcase.community/ (Accessed on: 2019-10-24)

the training or the test set constitutes an infeasible approach for the application of ALC, and the provided metadata does not allow to introduce a significant temporal gap between the training and the test data, the impracticality of enforcing generalization results in a high classification accuracy, but in an unreliable generalization estimate [Valenti2016].
Based on our proposed dataset, we investigated the impact of local as well as of temporal constraints for the application of ASC. Therefore, we observed a lower classification accuracy for local constraints compared to temporal ones. However, this behavior might be explained by the limited number of locations for each acoustic scene, hence, providing insufficient knowledge for the generalization of locations to acoustic scenes when locally constrained.
Finally, we investigated the application of ALC with temporal constraints on our proposed dataset. Therefore, introducing temporal constraints results in a decrease of the classification accuracy by $21,35\%$, from 85.17% without constraints to 63.82% with temporal constraints, but provides a reliable generalization estimate [Valenti2016] for the classification of audio samples to the location of origin. However, when we investigated the CV runs in detail, we observed a misclassification of 100% for a particular location. Thus, the limitations for ALC arise from the limitations of the underlying dataset. More precisely, our dataset provides sufficient data to estimate the generalization performance, but lacks in knowledge to generalize the acoustic representation of locations over a longer period of time. Summarized, introducing local and temporal constraints on ASC, and ALC, respectively, constitutes a prerequisite to reliably estimate the generalization performance [Valenti2016] and is fundamental to mitigate overfitting [Mesaros2018a].

5.1. Future Work

Due to the limited number of audio samples, i.e., 8 locations, and 3 days of collecting audio recordings, we suggest the augmentation of the proposed dataset by including more locations and extending the time frame to gather daily audio recordings over several weeks, or even months, while preserving a constant and restricted intraday time frame. Furthermore, artificial data augmentation, as proposed in [Chen2019], provides promising results. However, we favour to manually augment the proposed dataset to confirm the empirical outcomes w.r.t. the practicality and limitations of classifying audio samples to the location of origin.

Additionally, we suggest to introduce a Sound Event Localization and Detection (SELD) based methodology, as proposed in [Adavanne2018, Adavanne2019], for the classification of locations. Thus, to introduce the artificial detection, and localization, of specific events within audio samples. [Adavanne2018]
Based on the obtained information w.r.t. the presence, absence, and the sequence of events, we expect an improvement for the application of ALC.
Furthermore, since the process of collecting and labelling a sufficiently large number of audio samples constitutes a highly time-consuming task, we suggest to capitalize on Transfer Learning (TL), as proposed in [Pan2009]. Hence, utilizing on well-known problem statements and datasets by transferring the obtained knowledge to ALC w.r.t. the proposed dataset. [Pan2009]
Additionally, we are looking forward to investigate the impact of different recording devices, as showcased by the DCASE 2019 ASC "with mismatched recording devices" challenge, while collecting audio samples with our proposed Android application. Although we collect *UNPROCESSED*[2] audio recordings, and enforce consistent audio properties, we have not yet empirically investigated the discrepancies between different devices.
Furthermore, the classification of indoor locations, i.e., artificially differentiating rooms based on the acoustic environment, as proposed in [Tarzia2011], represents a promising field of future work. Thus, we suggest to establish and investigate a dataset which provides audio samples over a sufficiently long period of time of indoor locations within selected buildings.
Moreover, employing an online model for ASC, and ALC, respectively, e.g., proposed in [Imoto2016], might represent another future direction. Thus, enabling the training and classification of audio samples once recorded, or even while recording. [Imoto2016, Sahoo2017]
Finally, when comparing the results of the baseline classifier, proposed for the DCASE 2019 ASC challenge in [Mesaros2018], to the winner's results, proposed in [Chen2019], we look forward to optimize suitable classifiers, specifically targeting the complexity and properties of our proposed dataset for ALC.

[2] `https://developer.android.com/reference/android/media/MediaRecorder.AudioSource.html#UNPROCESSED` (Accessed on: 2019-10-24)

Bibliography

[Deng2009] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, 'ImageNet: A Large-Scale Hierarchical Image Database', in *CVPR09*, 2009.

[Mesaros2018] A. Mesaros, T. Heittola and T. Virtanen, 'A multi-device dataset for urban acoustic scene classification', in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, Nov. 2018, pp. 9–13. [Online]. Available: https://arxiv.org/abs/1807.09840.

[Valenti2016] M. Valenti, A. Diment, G. Parascandolo, S. Squartini and T. Virtanen, 'Dcase 2016 acoustic scene classification using convolutional neural networks', English, in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, Tampere University of Technology. Department of Signal Processing, Sep. 2016.

[Mesaros2016] A. Mesaros, T. Heittola and T. Virtanen, 'TUT database for acoustic scene classification and sound event detection', in *24th European Signal Processing Conference 2016 (EUSIPCO 2016)*, Budapest, Hungary, 2016.

[Legenstein2017] R. Legenstein, *Lecture notes for tu graz cs class 708.075: Neural networks*, 2017. [Online]. Available: https://igi-web.tugraz.at/lehre/NeuralNetworks/WS17/.

[Knoebelreiter2017] P. Knöbelreiter, *Lecture notes for tu graz cs class 710.034: Optimization for computer science*, 2017. [Online]. Available: https://tc.tugraz.at/main/enrol/index.php?id=1352.

Bibliography

[Goodfellow2016] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org. [Online]. Available: http://www.deeplearningbook.org/.

[Werbos1974] P. J. Werbos, 'Beyond regression: New tools for prediction and analysis in the behavioral sciences', PhD thesis, Harvard University, 1974.

[Rumelhart1986] D. E. Rumelhart, G. E. Hinton and R. J. Williams, 'Learning Representations by Back-propagating Errors', *Nature*, vol. 323, no. 6088, pp. 533–536, 1986. DOI: 10.1038/323533a0. [Online]. Available: http://www.nature.com/articles/323533a0.

[Fukushima1980] K. Fukushima, 'Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position', *Biological cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[LeCun1989] Y. LeCun *et al.*, 'Generalization and network design strategies', in *Connectionism in perspective*, vol. 19, Citeseer, 1989.

[Springenberg2014] J. T. Springenberg, A. Dosovitskiy, T. Brox and M. Riedmiller, 'Striving for simplicity: The all convolutional net', *arXiv preprint arXiv:1412.6806*, 2014.

[Karpathy2019] J. J. Andrej Karpathy, *Lecture notes for stanford cs class cs231n: Convolutional neural networks for visual recognition*, 2019. [Online]. Available: http://cs231n.github.io/convolutional-networks/.

[Srivastava2014] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, 'Dropout: A simple way to prevent neural networks from overfitting', *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[Ioffe2015] S. Ioffe and C. Szegedy, ‘Batch normalization: Accelerating deep network training by reducing internal covariate shift’, *CoRR*, vol. abs/1502.03167, 2015. arXiv: 1502.03167. [Online]. Available: http://arxiv.org/abs/1502.03167.

[Shimodaira2000] H. Shimodaira, ‘Improving predictive inference under covariate shift by weighting the log-likelihood function’, *Journal of statistical planning and inference*, vol. 90, no. 2, pp. 227–244, 2000.

[Prechelt1998] L. Prechelt, ‘Early stopping-but when?’, in *Neural Networks: Tricks of the trade*, Springer, 1998, pp. 55–69.

[Kingma2014] D. P. Kingma and J. Ba, ‘Adam: A method for stochastic optimization’, *arXiv preprint arXiv:1412.6980*, 2014.

[Duchi2011] J. Duchi, E. Hazan and Y. Singer, ‘Adaptive subgradient methods for online learning and stochastic optimization’, *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[Tieleman2012] T. Tieleman and G. Hinton, ‘Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude’, *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[Graves2013] A. Graves, ‘Generating sequences with recurrent neural networks’, *arXiv preprint arXiv:1308.0850*, 2013.

[Masters2018] D. Masters and C. Luschi, *Revisiting small batch training for deep neural networks*, 2018. arXiv: 1804.07612 [cs.LG].

[LeCun1998] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, ‘Gradient-based learning applied to document recognition’, *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[Polyak1964] B. Polyak, ‘Some methods of speeding up the convergence of iteration methods’, *Ussr Computational Mathematics and Mathematical Physics*, vol. 4, pp. 1–17, Dec. 1964. DOI: 10.1016/0041-5553(64)90137-5.

[Chollet2015] Chollet *et al.*, *Keras*, https://keras.io, 2015.

[Abadi2015] M. Abadi *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[Mesaros2018a] A. Mesaros, T. Heittola and T. Virtanen, 'Acoustic scene classification: An overview of dcase 2017 challenge entries', in *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*, IEEE, 2018, pp. 411–415.

[Schmidhofer2018] A. Schmidhofer, 'Dataset generation guideline for acoustic transport mode detection', Master's thesis, Graz University of Technology, 2018.

[Chen2019] H. Chen, Z. Liu, Z. Liu, P. Zhang and Y. Yan, 'Integrating the data augmentation scheme with various classifiers for acoustic scene modeling', DCASE2019 Challenge, Tech. Rep., Jun. 2019.

[Adavanne2018] S. Adavanne, A. Politis, J. Nikunen and T. Virtanen, 'Sound event localization and detection of overlapping sources using convolutional recurrent neural networks', *IEEE Journal of Selected Topics in Signal Processing*, pp. 1–1, 2018, ISSN: 1932-4553. DOI: 10.1109/JSTSP.2018.2885636.

[Adavanne2019] S. Adavanne, A. Politis and T. Virtanen, 'A multi-room reverberant dataset for sound event localization and uetection', in *Submitted to Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, 2019. [Online]. Available: https://arxiv.org/abs/1905.08546.

[Pan2009] S. J. Pan and Q. Yang, 'A survey on transfer learning', *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[Tarzia2011] S. P. Tarzia, P. A. Dinda, R. P. Dick and G. Memik, ‘Indoor localization without infrastructure using the acoustic background spectrum’, in *Proceedings of the 9th international conference on Mobile systems, applications, and services*, ACM, 2011, pp. 155–168.

[Imoto2016] K. Imoto and N. Ono, ‘Online acoustic scene analysis based on nonparametric bayesian model’, in *2016 24th European Signal Processing Conference (EUSIPCO)*, IEEE, 2016, pp. 988–992.

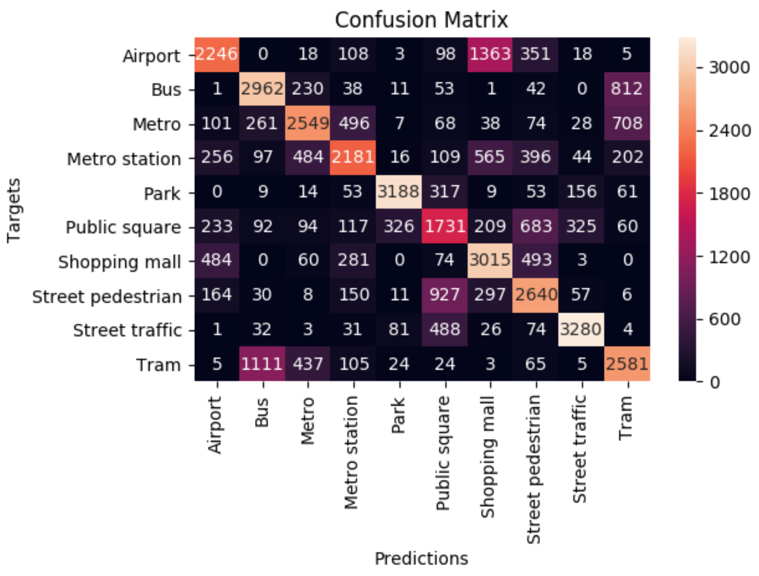
[Sahoo2017] D. Sahoo, Q. Pham, J. Lu and S. C. Hoi, ‘Online deep learning: Learning deep neural networks on the fly’, *arXiv preprint arXiv:1711.03705*, 2017.

Appendix

Appendix A. Experiments

A.1. TAU Acoustic Scene Classification Proof of Concept

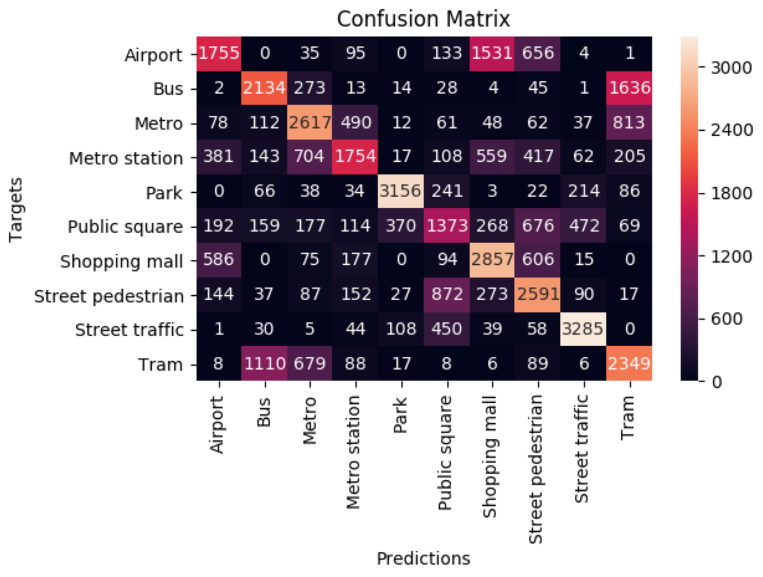
Figure A.1.: Confusion matrix, representing the accumulated results over 10 independent training runs of employing ASC on the suggested train-test split for the TAU development dataset.



Targets \ Predictions	Airport	Bus	Metro	Metro station	Park	Public square	Shopping mall	Street pedestrian	Street traffic	Tram
Airport	2246	0	18	108	3	98	1363	351	18	5
Bus	1	2962	230	38	11	53	1	42	0	812
Metro	101	261	2549	496	7	68	38	74	28	708
Metro station	256	97	484	2181	16	109	565	396	44	202
Park	0	9	14	53	3188	317	9	53	156	61
Public square	233	92	94	117	326	1731	209	683	325	60
Shopping mall	484	0	60	281	0	74	3015	493	3	0
Street pedestrian	164	30	8	150	11	927	297	2640	57	6
Street traffic	1	32	3	31	81	488	26	74	3280	4
Tram	5	1111	437	105	24	24	3	65	5	2581

A.2. TAU Acoustic Scene Classification

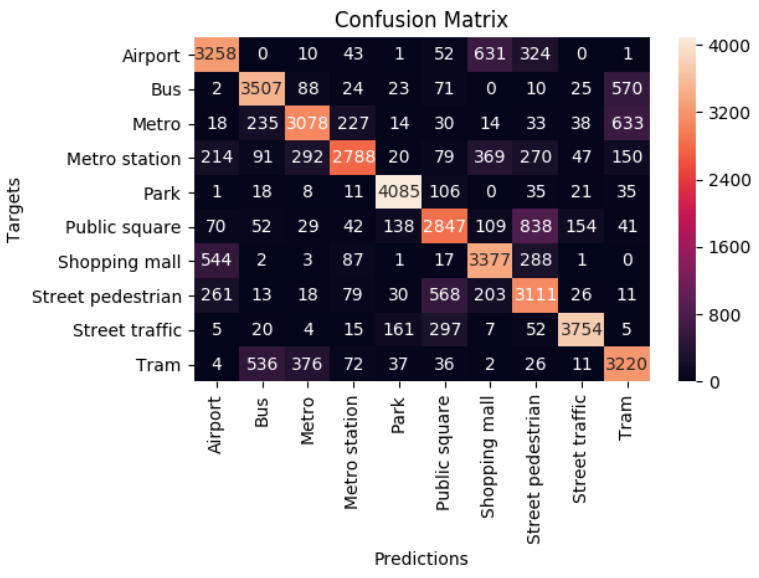
Figure A.2.: Confusion matrix, representing the accumulated results of 10 independent training runs for employing ASC on the TAU development dataset, while utilizing 10% of the training set for validation.



Targets \ Predictions	Airport	Bus	Metro	Metro station	Park	Public square	Shopping mall	Street pedestrian	Street traffic	Tram
Airport	1755	0	35	95	0	133	1531	656	4	1
Bus	2	2134	273	13	14	28	4	45	1	1636
Metro	78	112	2617	490	12	61	48	62	37	813
Metro station	381	143	704	1754	17	108	559	417	62	205
Park	0	66	38	34	3156	241	3	22	214	86
Public square	192	159	177	114	370	1373	268	676	472	69
Shopping mall	586	0	75	177	0	94	2857	606	15	0
Street pedestrian	144	37	87	152	27	872	273	2591	90	17
Street traffic	1	30	5	44	108	450	39	58	3285	0
Tram	8	1110	679	88	17	8	6	89	6	2349

A.3. TAU Acoustic Scene Classification Unconstrained

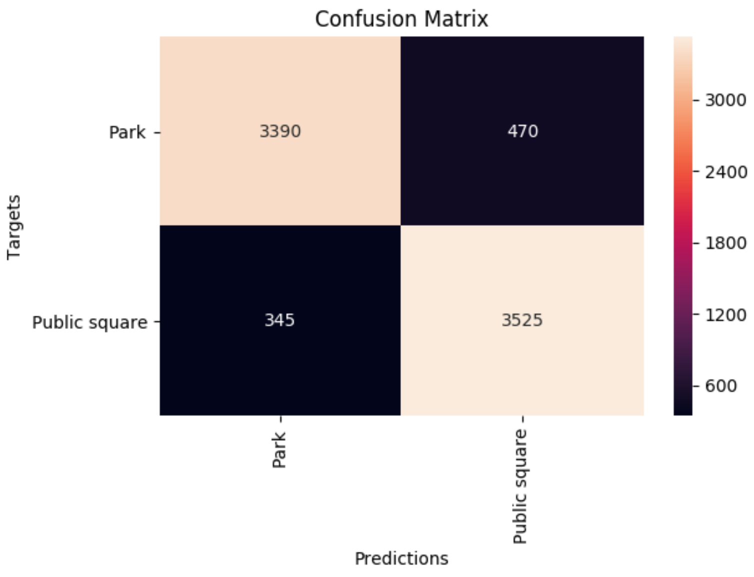
Figure A.3.: Confusion matrix, illustrating the accumulated results over 10 independent training runs of employing ASC on the TAU development dataset without the proposed constraints on the train-test split.



	Airport	Bus	Metro	Metro station	Park	Public square	Shopping mall	Street pedestrian	Street traffic	Tram
Airport	3258	0	10	43	1	52	631	324	0	1
Bus	2	3507	88	24	23	71	0	10	25	570
Metro	18	235	3078	227	14	30	14	33	38	633
Metro station	214	91	292	2788	20	79	369	270	47	150
Park	1	18	8	11	4085	106	0	35	21	35
Public square	70	52	29	42	138	2847	109	838	154	41
Shopping mall	544	2	3	87	1	17	3377	288	1	0
Street pedestrian	261	13	18	79	30	568	203	3111	26	11
Street traffic	5	20	4	15	161	297	7	52	3754	5
Tram	4	536	376	72	37	36	2	26	11	3220

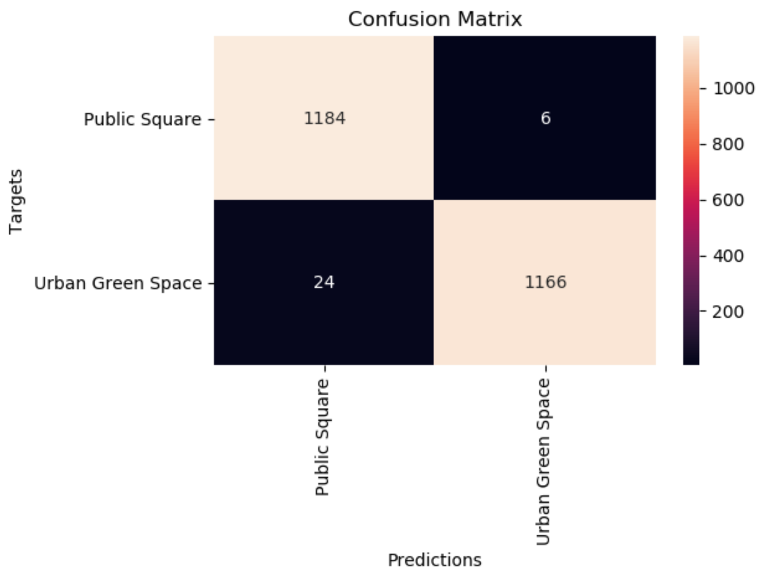
A.4. TAU Acoustic Scene Classification on Urban Parks & Public Squares

Figure A.4.: Confusion matrix, representing the accumulated results of 10 independent training runs of employing ASC, considering solely locations associated with *Urban Parks* and *Public Squares* provided by the TAU development dataset.



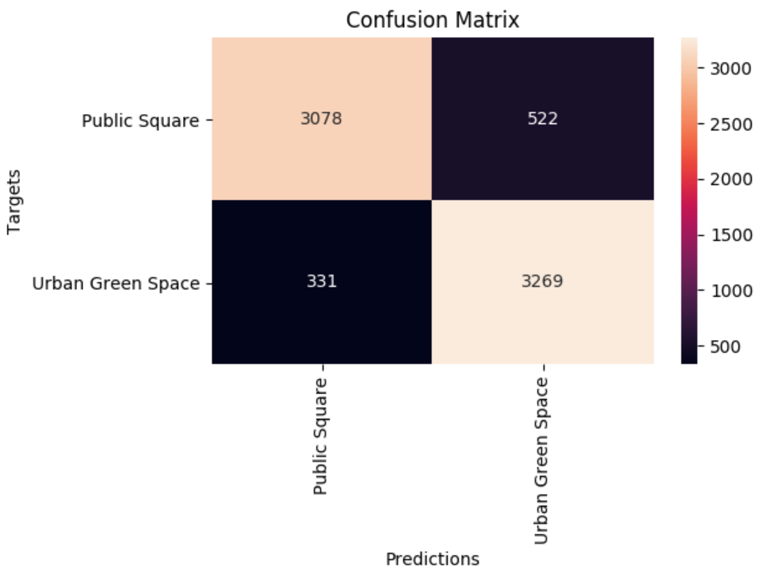
A.5. Proposed Acoustic Scene Classification

Figure A.5.: Confusion matrix, representing ASC without constraints on our proposed dataset, accumulated over 10 independent training runs. Training, validation, and test are set randomly split for each training run independently.



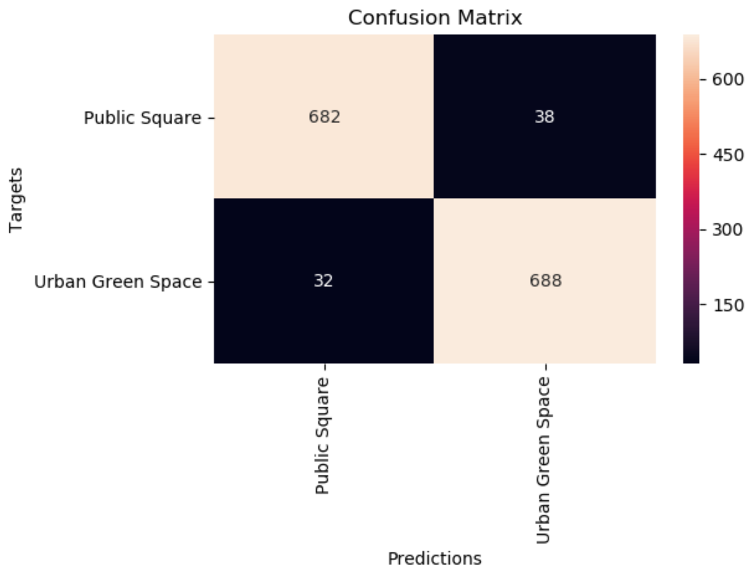
A.6. Proposed Acoustic Scene Classification with Local Constraints

Figure A.6.: Confusion matrix, representing the accumulated results of employing ASC with local constraints on our proposed dataset w.r.t. the approached CV presented in Figure 4.11 with 10 independent training runs per CV run.



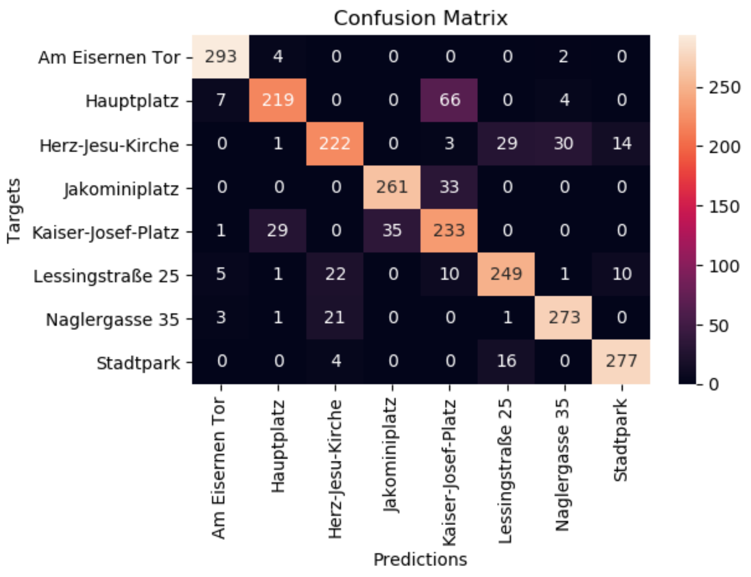
A.7. Proposed Acoustic Scene Classification with Temporal Constraints

Figure A.7.: Confusion matrix, representing the results of employing ASC with temporal constraints on our proposed dataset, hence constituting the outcome of employing a 3-fold CV w.r.t. the permutations. Thus, splitting the proposed dataset by the date of recording into 3 equally sized folds, and accumulating the outcomes over 6 independent training runs.



A.8. Proposed Acoustic Location Classification

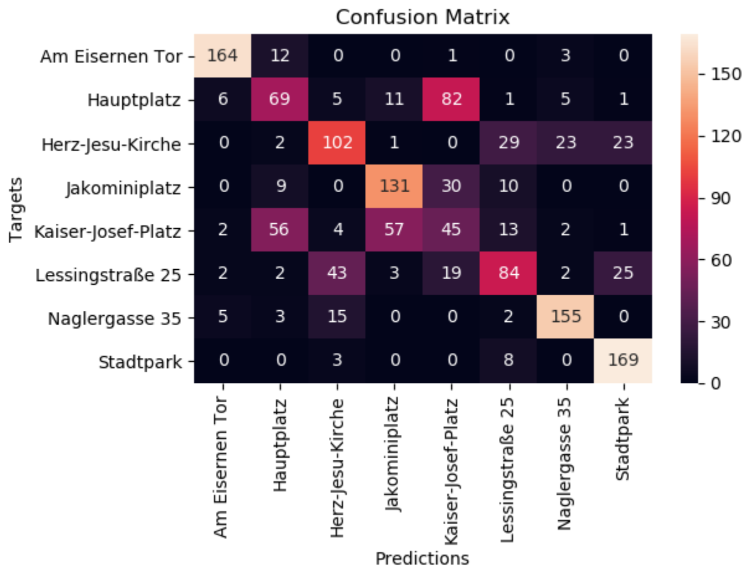
Figure A.8.: Confusion matrix, representing the outcomes of employing ALC without constraints on our proposed dataset, accumulated over 10 independent training runs, randomly splitting the dataset into training (33%), validation (33%), and test set (33%), independently for each run.



Targets \ Predictions	Am Eisernen Tor	Hauptplatz	Herz-Jesu-Kirche	Jakominiplatz	Kaiser-Josef-Platz	Lessingstraße 25	Naglergasse 35	Stadtpark
Am Eisernen Tor	293	4	0	0	0	0	2	0
Hauptplatz	7	219	0	0	66	0	4	0
Herz-Jesu-Kirche	0	1	222	0	3	29	30	14
Jakominiplatz	0	0	0	261	33	0	0	0
Kaiser-Josef-Platz	1	29	0	35	233	0	0	0
Lessingstraße 25	5	1	22	0	10	249	1	10
Naglergasse 35	3	1	21	0	0	1	273	0
Stadtpark	0	0	4	0	0	16	0	277

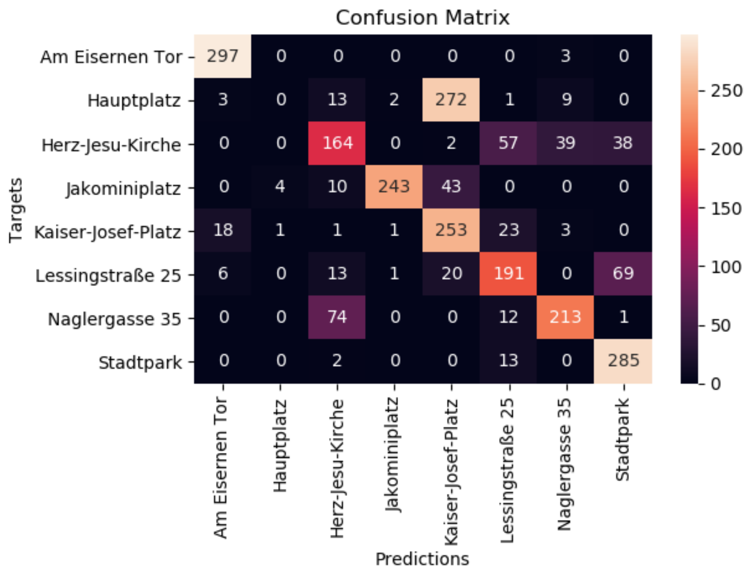
A.9. Proposed Acoustic Location Classification with Temporal Constraints

Figure A.9.: Confusion matrix, representing the accumulated outcomes of the 6 CV runs w.r.t. the permutations of the introduced 3-fold CV for ALC with temporal constraints on our proposed dataset.



Targets \ Predictions	Am Eisernen Tor	Hauptplatz	Herz-Jesu-Kirche	Jakominiplatz	Kaiser-Josef-Platz	Lessingstraße 25	Naglergasse 35	Stadtpark
Am Eisernen Tor	164	12	0	0	1	0	3	0
Hauptplatz	6	69	5	11	82	1	5	1
Herz-Jesu-Kirche	0	2	102	1	0	29	23	23
Jakominiplatz	0	9	0	131	30	10	0	0
Kaiser-Josef-Platz	2	56	4	57	45	13	2	1
Lessingstraße 25	2	2	43	3	19	84	2	25
Naglergasse 35	5	3	15	0	0	2	155	0
Stadtpark	0	0	3	0	0	8	0	169

Figure A.10.: Confusion matrix of 10 independent training runs w.r.t. the 1st CV run of the 3-fold CV for the evaluation of ALC with temporal constraints on our proposed dataset.



Targets \ Predictions	Am Eisernen Tor	Hauptplatz	Herz-Jesu-Kirche	Jakominiplatz	Kaiser-Josef-Platz	Lessingstraße 25	Naglergasse 35	Stadtpark
Am Eisernen Tor	297	0	0	0	0	0	3	0
Hauptplatz	3	0	13	2	272	1	9	0
Herz-Jesu-Kirche	0	0	164	0	2	57	39	38
Jakominiplatz	0	4	10	243	43	0	0	0
Kaiser-Josef-Platz	18	1	1	1	253	23	3	0
Lessingstraße 25	6	0	13	1	20	191	0	69
Naglergasse 35	0	0	74	0	0	12	213	1
Stadtpark	0	0	2	0	0	13	0	285