PROJECT 4: Acoustic Scene Classification

Elena Molinari

Melania Moras

Paolo Stella

Enrico Torres

1. Introduction
   1. Descripion of the task

The assigned task is Acoustic scene classification (ASC), whose purpose is to assign a label to an audio stream recorded in a certain environment by analyzing the audio signal. The data consists of 30-second audio files (WAV, stereo, 44.1 kHz, 16-bit), recorded using binaural headphones in locations around London at various times in 2012. In total, there are 200 audios equally distributed between ten different environments, which are: office, restaurant, bus, quiet street, busy street, tube, tube station, park, supermarket, open air market.

* 1. Characteristics of ASC

The dataset for this task exhibits a greater diversity and complexity with respect to speech or music signals. First, the length of the signals (30 seconds) suggests that in order to correctly identify the acoustic environment it is required to analyze long-term characteristics, in order to identify temporal patterns in the signals. For example, we can find many sound events over time or at the same time, and it is important not only to correctly identify them but also to analyze their characteristics. For example, we can find speech in both open air market and office, but it will be much quieter in the latter environment. Urban sounds are of course present in almost every signal, since they are all recorded in a city. We can assume that we will need a large number of features for the classification in order to capture the differences between each acoustic scene.

* 1. Literature review

ASC literature dates back to the 1970s, with the first work on the topic being soundscape (Schafer, 1977), pioneering work in which researchers pay attention to acoustic environments. Later, multiple ASC related concepts were proposed, and the automatic classification of environmental sounds received some attention. In the second phase, the publication of “computational auditory scene analysis” (Wang & Brown, 2006) and the study of comprehensively evaluating computers’ and humans’ perception of audio context (Eronen et al., 2006) started the early works of ASC. In this phase, ASC algorithms applied features from speech recognition areas and simple machine learning methods, as well as several ASC datasets were released. The third phase started from the DCASE2016 challenge (Eghbal-Zadeh et al., 2016), where deep learning methods were introduced into ASC and achieved high performance. In this phase, ASC received widespread attention, and the ASC publications rapidly increased. The performance of the research obtained some remarkable results on the LITIS dataset, with an accuracy of 98,9% in 2021.

Early methods were mainly based on MFCC features and simple machine learning methods, such as random forest, MLP, and SVM. Recently, features like LMBE, spectrogram, CQT, and log Mel spectrogram (or their variants) were adopted with CNN-based classifiers, such as ResNet, FCNN, and Xception. Feature learning combined with a simple classifier (e.g., MLP) also performs well in some datasets. Furthermore, most high-performance approaches employ data augmentation and late fusion methods, especially for processing complex datasets (e.g., D17 to D23).

Although the ASC accuracy on early datasets has been significantly improved to over 95 %, the accuracy on recent datasets is still low (less than 90 %), such as D20A, D21A, D22, and D23 datasets recorded by multiple devices. This is also due to the increase of task complexity (e.g., larger-scale data, multiple devices, limited model complexity, and shorter segments). Compared with the D20A task, D21A only added a low-complexity constraint, slightly decreasing accuracy.

* 1. Problems

Although the ASC field has made great progress, there are also some problems that need to be addressed. We analyze them from the perspectives of acoustic scenes, datasets, classification performance, and deployment in the real world

1. ***Unclear definition of acoustic scenes***. ASC requires a clear definition of acoustic scenes, as it needs predefined labels. However, it is hard to define a generalized boundary for acoustic scenes, since the acoustic signal is complex, multi-source, overlapping, and unstructured. Furthermore, the distribution of sound source components varies from application-to-application scenarios. For example, human activities and traffic are very intensive in urban environments, distinct from rural environments.

2. ***Lack of large-scale comprehensive datasets***. As a complex multi-classification task, a high-performance ASC system requires a large-scale dataset for model training. However, the size and diversity of the available data are currently limited. Large-scale comprehensive datasets are lacking for developing a specific ASC system, despite many existing ASC datasets.

3. ***Low classification performance***. Compared with other audio fields, the ASC’s performance is still deficient. There are mainly three reasons: (i) The high diversity among acoustic features of the same scenes and the similarity among acoustic features of different scenes make it difficult to extract efficient features for ASC, (ii) Audio signals lack structural information, making modeling difficult compared with speech signals, and (iii) Mismatched data distribution between source and target data, such as multiple devices, might cause low performance.

4. ***Computational efficiency and practicability***. At present, ASC models with high performance are relatively complex. There are problems with computational efficiency optimization and algorithm adaptability when deploying ASC algorithms in the real world. For example, in the reality of embedded industrial applications, an ASC system might be deployed in conditions of limited computational power and memory capacity (e.g., IoT embedded devices), where the high performance of the ASC system is also required. Therefore, the computational efficiency of the ASC system needs to be considered to judge whether it is practical for this application.

1. Methodology
   1. Initial steps

We worked with both the datasets available on the website, since the second one was initially private and then made public for testing. We merged the two folders into a single dataset, deciding to keep 50 samples for testing and perform data augmentation on the other 150, making them our training set.

First, we plotted the signals in order to gain familiarity with them, and we also plotted their Mel Spectrogram, in order to find differences between them. As expected, loud and quiet scenes have visible differences, while loud environments such as tube and bus do not have visible peculiarities. This confirms the hypothesis that using a small number of features will not result in accurate performances.

* 1. Preprocessing

The first preprocessing step is data augmentation. The data augmentation techniques that we employed are: time stretch, time compression, pitch shifting (up or down). In order to have the same number of samples per signals also for the augmented files, we had two options for the time stretched/compressed files, which were: zero padding and truncation or resampling. We opted for the latter, and we performed the four augmentation for each of the train files. We discovered soon that the stretched and compressed data compromised the results of the classification, so we eventually decided to discard them, keeping only the original files along with their pitch shifted versions.

Then we applied a scaler onto the signals, in order to normalize them.

* 1. Feature extraction

As a first approach, we created a feature vector containing the mean and standard deviation of the following quantities:

1. Zero-crossing rate
2. Mel spectrogram
3. Derivative of the Mel spectrogram
4. Second derivative of the Mel spectrogram
5. Spectral flux
6. MFCC
   1. Model

We opted for an SVM model, since previous reasearches in the literature seemed to obtain good results with it. The best kernel resulted to be ‘linear’, and the best value of C is 100. We trained the model on the extracted feature vectors and evaluated its performance on both the training and test sets. Here we summarize our results:

* With the time-stretched signals the model was overfitting, so we discarded them as aforementioned.
* Without them the performance was better but still overfitting, so we discarded the MFCC features, obtaining a train accuracy og 0,8 and a test accuracy of 0,72.
* Changing the values for the computation to n = 2048 and h = 256 we had a train accuracy of 0,7 and a test accuracy of 0,74.
* We then decided to add more features in order to improve the performance

Final configuration of features:

Mean and standard deviation of:

1. Zero-crossing rate
2. Mel spectrogram
3. Derivative of the Mel spectrogram
4. Second derivative of the Mel spectrogram
5. Spectral flux
6. Spectral contrast
7. Tonnetz
8. Pcen

The model achieved an accuracy of 78.2% on the training data and 74% on the test data, indicating a good generalization capability with only a moderate gap between training and testing accuracy. This demonstrates that the SVM with a linear kernel is effective in capturing the patterns in our acoustic scene features, providing a reliable baseline for further improvements.

2.5 Confusion Matrix

To further analyze the performance of the SVM model, we evaluated the confusion matrix on the training data. This allowed us to observe which classes the model is able to correctly distinguish and where it makes the most mistakes. Along with the confusion matrix, we also generated a classification report, which provides more detailed metrics such as precision, recall, and F1-score for each class. These tools offer a deeper insight compared to overall accuracy alone, helping us identify potential imbalances or weaknesses in the classifier’s behavior.

In particular, precision measures the proportion of correct positive predictions among all positive predictions, helping us understand how reliable the model is when it predicts a specific class. Recall instead measures the proportion of actual positives that were correctly identified, which tells us how sensitive the model is in detecting each class. The F1-score combines both precision and recall into a single metric, which is especially useful when the dataset is imbalanced. Finally, the support value indicates how many actual instances of each class are present in the dataset, giving context to the other metrics.

By analyzing these values, we can better interpret the classifier's strengths and weaknesses, beyond what accuracy alone can tell us.

We can observe that the model performs well on some classes such as bus, restaurant, open air market, busy street, supermarket. On the other hand, there is some confusion between park and quietstreet, and we can assume that this is because they are both quiet environments without much background noise.

After assessing the performance on the training data, we extended the same analysis to the **test set**. In this phase, we computed the confusion matrix and classification report on the predictions obtained from unseen data. This comparison is crucial to evaluate the model’s generalization capability. If the performance metrics remain stable across both training and test sets, it suggests that the model is robust and not overfitting. On the other hand, a significant drop in precision, recall, or F1-score for some classes might reveal difficulties in handling unseen samples or indicate class-specific weaknesses.

The report of the test set show that the model fails to identify the class tube station, and has some problems with supermarket and restaurant as well. Overall, the performance is not very different from the one on the train set, since the wrong classifications (false positives and false negatives) happen between the same classes for both the datasets.

* 1. Compare model

To compare the SVM with other traditional machine learning models, we also evaluated the performance of a k-Nearest Neighbors (kNN) classifier. We set the number of neighbors to k=10 and trained the model using the same feature vectors used for the SVM. However, the results were unsatisfactory: the model showed rather low accuracy on the training data, indicating difficulties in capturing meaningful patterns. Performance further deteriorated on the test set, confirming that the model was not suitable for our problem or the type of data used.

* 1. Prepare to CNN

To prepare the data for training a convolutional neural network (CNN), we converted the audio signals into Mel spectrograms, a time-frequency representation that captures the most relevant acoustic features for recognizing sound patterns. Mel spectrograms are particularly effective at representing human auditory perception, making them an ideal choice for deep learning models in audio applications and we also checked if the length of it is the same for all the data otherwise we zeropad it. After that we compute the dB conversion, since the dB scale is closer to human perception and we normalized the result obtained.

Once the spectrograms were obtained, we organized them into multidimensional arrays, adding a “channel” dimension to fit the input structure required by CNNs, which typically process image-like data with multiple channels. To maintain consistency across the dataset, we standardized the temporal length of the spectrograms by cropping them to a fixed size. This step is crucial to ensure that all instances have the same shape, a necessary condition for neural network training.

At the same time, class labels were encoded using one-hot encoding, transforming nominal categories into binary vectors. This representation facilitates multi-class classification during training, enabling the network to learn more effectively.

We applied the same process to the test data, ensuring that transformations and dimensions were consistent with those of the training set. In summary, this workflow allowed us to prepare a structured and ready-to-train dataset for a CNN, optimizing feature quality and compatibility with the chosen architecture.

* 1. 2D-Convolutional Neural Network

The neural network that we implemented is composed of 5 hidden layers. Four of them are a sequence of Conv2D (where the number of filter double for each layer starting from 32), Max pooling 2D and Batch normalization while the last one, after a global average pooling 2D is a Dense layer with activation = ‘relu’. The output stage is a Dense layer with 10 neurons and an activation=’softmax’. We compile the model using an Adam optimizer with a learning rate of 0.00001.

Then we train the model running it for 500 epochs and when we evaluate the model on the test data we reach an accuracy of 68%. This result shows a level of criticality due to the small amount of data that we used. So if we had at our disposal a bigger dataset we could reach a similar result even with a smaller number of epochs reducing its computational load.

To see the source code and images:

<https://github.com/enry1897/Selected_P4/tree/main>

(The code was executed using Kaggle: <https://www.kaggle.com/code/enry1897/selecte-p4/edit>)