TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

VILLE-VEIKKO EKLUND

AUDIO DATASET CREATION

Bachelor of Science thesis

Supervisor: M. Sc. Aleksandr Diment
Submitted for examination on
8th December 2017

ABSTRACT

VILLE-VEIKKO EKLUND: Audio dataset creation
Tampere University of Technology
Bachelor of Science thesis, 16 pages
December 2017
Bachelor's Degree Programme in Computing and Electrical Engineering
Major: Signal Processing
Supervisor: M. Sc. Aleksandr Diment
Keywords: environmental audio, audio dataset, Freesound, supervised learning

The goal of this thesis was to build an easily accessable and approachable environmental audio dataset for benchmarking machine learning algorithms. The samples in the created dataset were collected from Freesound, which is an online database consisting of sounds uploaded by its users. The dataset consists of three classes, each containing 50 instances of original recordings with annotations for 5-second segments. The classes are 'dog', 'bird' and 'rain', and the dataset was therefore given the name DBR dataset. Additionally, a script for evaluating support vector, random forest and k-NN classifiers with the dataset was written. The building and evaluation stages were documented step-by-step and a Jupyter notebook tutorial for using the dataset was created. The dataset, the scripts and the notebook were published online.

PREFACE

This thesis was written for the Bachelor's Seminar in Signal Processing during fall 2017. I would like to thank Aleksandr Diment for the guidance and feedback during the project. I would also like to thank Joni Kämäräinen for organizing the seminar, Toni Heittola for providing the thesis topic and Tuomas Virtanen for introducing me to the right people.

Tampere, 8.12.2017

Ville-Veikko Eklund

CONTENTS

LIST OF ABBREVIATIONS AND SYMBOLS

API	Application Protocol Interface
CC license	Creative Commons license
k-NN	k-Nearest Neighbors
SVC	Support Vector Classifier
SVM	Support Vector Machine
MFCC	Mel Frequency Cepstral Coefficient

1. INTRODUCTION

Supervised learning is a machine learning scenario, where a machine makes predictions on unseen data based on labeled examples. The process of supervised learning consists of five main steps: acquisition of data, labeling the data, extracting features, training a classifier, and evaluating the classifier. Datasets in supervised learning are complete collections of labeled examples intended for evaluating classification methods.

There are several datasets for machine learning tasks distributed over the Internet. These datasets consist of images, sound clips or anything that can be easily classified. For example, in image processing there is ImageNet [4], audio processing has AudioSet [9], and in emotion recognition there is Recola [16].

At the moment, the field of audio processing suffers from the lack of a standardized dataset, like CIFAR[1] in image processing. Furthermore, there are only a handful of open and precisely labeled datasets available due to licensing and the amount of work labeling requires.

The goal of this thesis is therefore to introduce a new, although concise, dataset to address the problem. The dataset consists of environmental audio samples that are collected from an online audio database Freesound [8]. After acquisition the data is manually labeled and the functionality of the set is then tested with popular classification algorithms. The code written for building and evaluating the dataset is distributed with the dataset. A Jupyter notebook tutorial for testing classifiers with the dataset is also released to make audio datasets more approachable for people unfamiliar with audio processing.

In Chapter 2, the available datasets and the tools used in building and evaluation are reviewed. Detailed documentation of the dataset building process is found in Chapter 3. Chapter 4 covers the evaluation of the dataset, and conclusions and further discussion are in Chapter 5.

[1] `https://www.cs.toronto.edu/~kriz/cifar.html`

2. BACKGROUND AND TOOLS

In this chapter, the state of environmental audio datasets and the tools used in dataset building and evaluation are reviewed.

2.1 Datasets

A dataset in supervised classification consists of data records, examples, with special target attributes called class labels. Classes are comprised of examples with similar attributes and same class labels. A classic example of a dataset is the Iris flower data set [7], which consists of 3 classes of different types of iris plants, each class containing attributes of 50 instances.

Audio signals are usually divided into three categories: music, speech and environmental audio. Environmental audio is the large group of sounds and audio events related to the environment as its name suggests, that is they are not recorded in a laboratory. Such sounds are for instance noises of animals, traffic noises and sounds related to weather phenomena. An environmental audio dataset can therefore be a collection of recordings of dogs barking or glass breaking for example.

Publicly available environmental audio datasets and their properties are listed in Table 2.1. According to the table it is evident that large and open datasets are somewhat difficult to find. A decreasing factor in the number of open datasets is the restriction of datasets to research purposes, which means accessing them requires authorization and their use in commercial applications is limited. The scarcity has recently inspired the creation of datasets such as UrbanSound, ESC and AudioSet. However, none of the available datasets has yet become a standard in the field, which makes baseline classifier comparison difficult.

The UrbanSound dataset by Salamon et al. from 2014 [19] was created in response to the lack of large annotated data but also to the missing common taxonomy in audio datasets. It used Freesound database as its source of data and as a result the Urban Sound Taxonomy and the largest free dataset of urban sounds was created. The samples in the dataset are original recordings with durations ranging from 1 s to

30 s. Another dataset, the UrbanSound8K, was constructed with a constant sample length of 4 seconds. The original code used in building and evaluating the datasets is not publicly available.

Table 2.1 *Available environmental audio datasets [21].*

Dataset name	Type	Classes	Examples	Size (min)
Sound Scenes				
Dares G1	recorded	28	123	123
DCASE 2013 Scenes	recorded	10	100	50
LITIS Rouen	recorded	19	3026	1513
TUT Sound Scenes 2016	recorded	15	1170	585
Environmental Sounds				
ESC-10	collected	10	400	33
ESC-50	collected	50	2000	166
NYU Urban Sound8K	collected	10	8732	525
CHIME-Home	recorded	7	6137	409
Freefield 1010	collected	7	400	33
CICESE Sound Events	collected	20	1367	92
AudioSet	collected	632	>2 mil	> 340k
Sound Events				
Dares G1	recorded	761	3214	123
DCASE 2013 Office Live	recorded	16	320	19
DCASE 2013 Office Synthetic	recorded	16	320	19
TUT Sound Events 2016	recorded	18	954	78
TUT Sound Events 2017	recorded	6	729	92
NYU Urban Sound	collected	10	3075	1620
TU Dortmund Multichannel	recorded	15	1170	585

Yet another dataset built on Freesound samples is the ESC dataset by Karol J. Piczak in 2015 [15]. The ESC-50 dataset consists of 50 classes of 5-second-long labeled environmental sounds, and the ESC-10 is simply a subset of the ESC-50. A bigger dataset consisting of 250 000 Freesound samples called ESC-US was also created with more or less weak automatic labels drawn from sample metadata. Piczak put more weight on the accessibility of original code and ease of use for the dataset, which is a step forward from UrbanSound in terms of openness. The Jupyter notebook released with the dataset contains also spectrograms of samples and a large amount of analyses.

It is notable that AudioSet [9] with its over 2 million examples and 632 classes is far bigger than all the other datasets shown in the table. AudioSet, created by Google in 2017, consists of samples from Youtube videos' audio tracks. Although AudioSet is easily accessible, the downloadable dataset contains only the pre-extracted features and no original audio. There are also problems with labeling accuracy: All the

original samples in AudioSet are 10-second clips containing a target sound with a class dependent quality estimate, which for a large number of classes is closer to 0 % than 100 % [1]. Furthermore, because the examples in the dataset are taken from Youtube videos, the dataset is vulnerable for content deletion.

It is worth mentioning also the DCASE (Detection and Classification of Acoustic Scenes and Events) datasets, which have been created for three DCASE challenges in 2013, 2016 and 2017. The first DCASE challenge was organized by researchers in Queen Mary University of London to stimulate research in machine listening, and the following challenges were held by Tampere University of Technology. The datasets used in the challenges consist of audio scene recordings in specific locations. For example, the TUT Sound Events 2017 dataset [14] contains 24 audio recordings from a single acoustic scene.

2.2 Dataset building

Dataset building consists of the selection of source of data, collecting, and labeling the data. Important requirements for the source are suitable licensing options for the samples and the ease of searching and retrieving data. Freesound was chosen as the source of samples because it satisfies both of these requirements, which is explained more thoroughly in section 2.3.

Building a dataset requires also a tool for labeling the samples. Labeling is the process of assigning class labels to examples in a dataset. With audio files, this involves inspecting the waveforms, selecting suitable segments, annotating them and saving the starting and ending times. This process can be done with Audacity[1], which is a free audio recording and editing software.

2.3 Freesound

Freesound is an online sound database designed for research and artistic purposes. The sounds in the database are recorded by Freesound's users, and the sounds are rated and commented by Freesound's active community. Since an application protocol interface[2] (API) was introduced in 2011, Freesound has been the source of data in a wide range of publications [11, 12, 18, 17]. With Freesounds free client libraries [3] designed for several programming languages searching and downloading

[1] `http://www.audacityteam.org/`
[2] `http://freesound.org/docs/api/index.html`
[3] `https://freesound.org/docs/api/client_libs.html`

samples is simple and efficient. The Python client was used in this project, which is explained in Chapter 3.

Samples in Freesound are licensed under four different kinds of Creative Commons (CC) licenses. The license types are:

- CC Sampling Plus 1.0[4]
- CC0 1.0[5]
- CC BY 3.0[6]
- CC BY-NC 3.0[7]

All of these licenses allow sampling and redistribution, which is desirable for dataset building. It is notable, that CC BY 3.0 license requires attribution to the author, and CC BY-NC 3.0 and Sampling Plus 1.0 licenses forbid commercial use in addition. CC0 1.0 license is the most allowing since it does not require any action from the user.

2.4 Dataset evaluation

The validity and applicability of a dataset can be confirmed by developing a machine learning application. Developing such an application requires tools for data preprocessing, extracting features and providing classifiers.

Preprocessing of data means optimising the data somehow for feature extraction, which in turn is the act of reducing the amount of data while preserving relevant information. For example, audio samples can be preprocessed by segmenting them into frames, which are easier to handle than the original waveforms. Features, such as Mel-frequency cepstral coefficients (MFCCs) [13], can then be extracted from the frames. MFCCs are coefficients derived from a mel-scaled short term power spectrum of a sound, and they are often used in speech recognition and music information retrieval. The mel scale is a widely used model of human frequency perception [20]. Preprocessing and feature extraction can be done with librosa[8], which is a Python package designed for music and audio signal processing.

[4] https://creativecommons.org/licenses/sampling+/1.0/
[5] https://creativecommons.org/publicdomain/zero/1.0/
[6] https://creativecommons.org/licenses/by/3.0/
[7] https://creativecommons.org/licenses/by-nc/3.0/
[8] https://github.com/librosa/librosa

Scikit-learn[9] is a machine learning module for Python, which provides classifiers for machine learning tasks. Choices for classifiers include support vector machine classifier (SVM classifier), random forest classifier and k-nearest neighbor classifier.

The standard support vector machines (SVMs) algorithm [3] is a non-probabilistic binary classifier, which divides the training set examples, as points in a plane, into distinct groups by a gap as wide as possible. To make the class separation possible, a high-dimensional feature space may be used instead of the space, where the problem was originally stated.

Random forest [10] is another popular algorithm used in classification and regression problems. It is a set of decision tree classifiers, which are tree-like structures consisting of nodes with test questions about the attributes of a sample. The answers of each question lead to the most fitting class. Predictions of individual trees are collected and the prediction with highest votes gets chosen as the final predicted class of the random forest. Because overfitting is not intrinsic for random forests, increasing the number of trees improves prediction accuracy.

K-nearest neighbors (k-NN) [2] is a simple non-parametric machine learning algorithm, whose output is dependent only on the k closest examples in the feature space. The class with the highest number of instances among the neighbors becomes the prediction.

An important step before training a classifier is to standardize the features of the data, because it is expected by many classifier algorithms. Standardizing means processing the features such that they become standard normally distributed data with zero mean and unit variance. Scikit-learn offers the class `StandardScaler()`, which takes care of the standardization.

Evaluation, the process of measuring a trained classifier's prediction accuracy, works as a measure of the dataset's validity. Evaluation methods differ in the way the dataset is split into training and test sets. One of them is Monte Carlo cross-validation, also known as repeated random subsampling, where the data is split randomly several times into training and test sets. The final prediction accuracy is the mean of accuracies for each split. [5]. For a three-class balanced scenario a suitable goal for average prediction accuracies would be 80–90 %, because it implies the classification problem is sufficiently hard but still solvable.

[9] `https://github.com/scikit-learn/scikit-learn`

3. BUILDING

This chapter focuses on the implementation of the data retrieval code and the process of selecting and labeling the samples. The script is available in the public repository of the dataset[1].

The constructed audio dataset has the following specifications:

- three classes: 'dog', 'bird', and 'rain'
- 50 samples per class
- sample length five seconds
- samples are continuous segments of the original sound

The number of classes in the dataset was limited to three and the number of samples per class to 50 in order for the dataset to be finished in the time constraints of this work, but also that the dataset would be big enough for testing. The duration of samples was set to five seconds so that a sufficient amount of data per sound would be gathered. The classes were chosen to be 'dog', 'bird' and 'rain' because searching sounds related to them yielded a high amount of results. Lastly, the use of continuous segments also reduces the workload.

The sound samples in this work were downloaded from Freesound using their free Python client[2], which was developed for the Freesound API. The Freesound API documentation thoroughly covers the usage of the API, and the client comes with an example file consisting of different types of queries and analyses on samples. Freesound.py in the client library offers all the needed functions and classes to query, filter and download sounds. However, to be able to download the original high quality sound files, Freesound requires Oauth2 verification. Unfortunately, the official client does not implement Oauth2 in it, but there is an extension library available[3] that

[1] `https://github.com/vvekl/dbr-dataset`
[2] `https://github.com/MTG/freesound-python`
[3] `https://github.com/xavierfav/freesound-python-tools`

takes care of the verification. The file manager.py contains the improved class Client, which overrides the official class.

The first step in downloading files from Freesound is applying for API keys. Registration to Freesound is required to be able to apply for the keys. Once an account is set up, the keys can be applied for on Freesound's website in the Developers section[4]. The application consists of basic information and the keys are awarded instantly after submission.

Connecting to Freesound API is done by creating a Client object.

```
client = manager.Client()
```

The client asks for the API keys and the user is directed to Freesound's webpage where the connection is established. The keys are asked for only on the first time connecting to Freesound API with Python, and they are saved in a .py-file automatically for subsequent reconnections.

The searching with the client is done with the `text_search()` -method found in the file freesound.py. The method has several parameters, but the most important ones are 'query', 'filter' and 'fields'. Query is the keyword to be searched, filters specify the search, and fields define what metadata is saved for the results. The method returns a pager object that lists the results of the search, 15 results per page by default. These results are sound objects, whose attributes are the metadata fields specified in the search.

The following is an example of a search for rain samples with results saved to a pager object 'results_pager'.

```
results_pager = client.text_search(
        query="raining",
        filter="duration:[10 TO 20] samplerate:[44100 TO *] type:wav",
        fields="id,name,description,duration,samplerate,username,license"
    )
```

The classes chosen in the dataset were 'dog', 'bird', and 'rain', and the keywords used in searching were 'dog barking', 'bird chirping' and 'raining' respectively. Printing out the metadata after a search proved useful in determining whether the used keyword was suitable. The field parameters used were 'id', 'name', description', duration', samplerate', 'username', and 'license'. The filters were only used to control the samplerate, sample length and type. The minimum samplerate was set to 44100

[4] `https://freesound.org/help/developers/`

Hz and the type to wav to filter out low quality samples.

The duration of a sample in the final dataset was set to five seconds. Therefore, the durations of the searched samples needed to be at least 10 seconds to make it possible to choose a five-second continuous segment with the target sound. The goal amount of samples per class in this work was 50, so it was optimal to reach for approximately 250 results in the search. This is due to high frequency of samples in Freesound with too much background noise and samples that have misleading titles. The upper limit of the duration can be used to limit the number of results effectively.

The sounds were downloaded with the sound object method `retrieve()` in freesound.py. The method's first parameter is the download directory and the second is the file name. If a filename is not specified, the sound attribute 'name' will be used instead. The files were named after their unique Freesound id's to prevent overlapping with names. To avoid mixing, samples of different classes were downloaded to separate folders. During the downloading process, the metadata, i.e. id, name, description, duration, samplerate, username and license, was saved in a dictionary format into a yaml file to assist with the process of choosing samples later.

When the downloading was complete, the types of licensing used with the samples were checked. This was done by extracting all the metadata dictionaries in the yaml files in to a list, then selecting only the license fields in to another list, and finally looking for unique entries in the license list. The samples in the dataset use all four types of licenses listed in section 2.3, and therefore the dataset can not be used for commercial purposes.

The id and the corresponding username fields of the sounds were also saved into class specific csv files. This makes it possible to keep track and control the number of sounds from the same user in the dataset. In the testing phase, when samples are split to training and test sets, the same user's samples must be restricted to only one of the two sets to pass cross-validation.

Next, the listening process and choosing of samples began. It took on average three rounds of listening to gather the required 50 samples. On the first round, if a sample clearly contained the target sound, it had little to none background noise and the description of the sound proved it real, an x was marked in the csv file on the row with the matching id to keep track of suitable samples. The following rounds consisted of more careful listening to reach the 50 sample mark, and only in the case of dog samples it was possible to leave out excess samples.

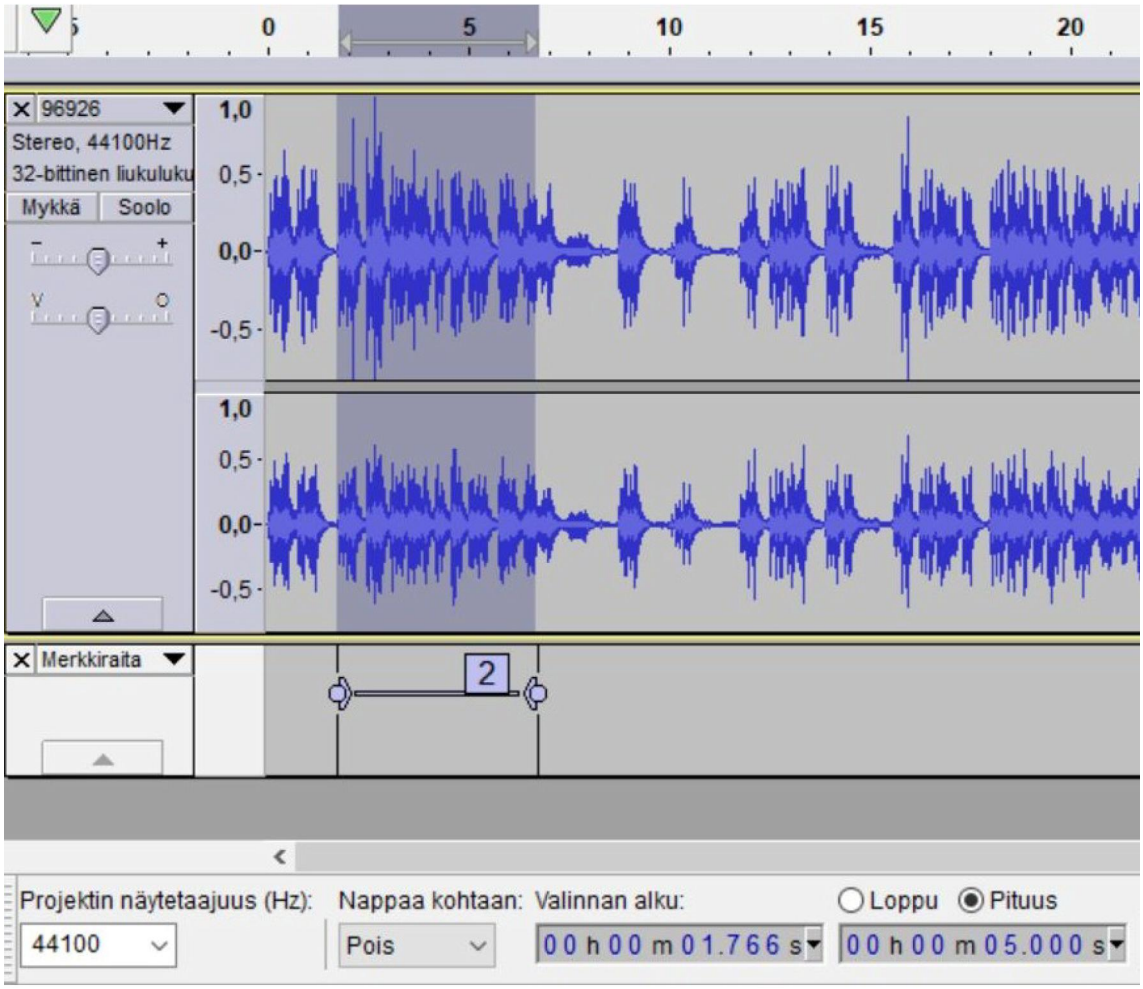

Figure 3.1 *A segment is chosen from a bird sample and marked with the class tag 2.*

The samples in this dataset were set to be five-second consistent clips without cutting the silent parts. Consequently, the starting time point, offset, had to be chosen and saved for later use. This was done with Audacity (Figure 3.1) by marking the starting and ending points (although the ending is not needed due to the fixed length) with a class tag and saving the marks into a text file with the same id as the sound. The tags for different classes in this work were as follows: '1' for dog, '2' for bird and '3' for rain. Including the class tag in the time offset files takes also care of labeling.

After this procedure, there were three folders for three classes, each containing 50 sounds, 50 metadata yaml files and 50 time offset/class tag text files. The next step was to test the dataset with a machine learning application, which is covered in chapter 4.

4. EVALUATION

In this chapter, the implementation of the machine listening application is reviewed and the evaluation results are discussed. The evaluation code is also available in the public repository of the dataset[1].

First, a list containing the ids of all samples was created, and time offsets and class tags were read into lists from the text files created with Audacity. A list containing full file paths was also created to make processing files more efficient. The wav files were loaded into a matrix according to the offsets with librosa, and the sample rates of the samples were saved into another list. The MFCCs were then extracted with librosa and the means of each set of features were saved into a feature list.

At this point, numpy random state generator was seeded and a RandomState was generated. Thus splitting, which includes shuffling the samples with numpy, would be different every round and the classifiers would use the same RandomState while supporting reproducibility of the results.

```
np.random.seed()
rnd = np.random.RandomState(42)
```

Next, all the examples in the dataset were split into training and test sets. To maintain same proportions of classes in both sets, the examples were split inside their respective classes.

The split process went as follows: All the unique usernames were collected from sample metadata and shuffled, and a list of sample ids was created in the order of the shuffled usernames. Next, the samples were split into 0.70/0.30 sets and the borderline samples' username fields were compared to ensure there was no flowing of same user's sounds into both sets. If a user had sounds in both sets, the split was done again until both sets had unique usernames. Once all the classes had been split into their own training and test sets, the sets were combined into one training and one test set.

[1] https://github.com/vvekl/dbr-dataset

After the split, the feature data and the corresponding class tags were collected for both sets from the lists created in the beginning. Lastly, the resulted feature lists were scaled with scikit-learn's `StandardScaler()`.

The classifiers used for evaluation were scikit-learn's `SVC()`, `RandomForestClassifier()` and `KNeighborsClassifier()`. They were initialized with `random_state` as rnd as defined above for SVM and random forest classifiers, penalty parameter C as 0.1 and kernel type 'linear' for SVM, number of decision trees for random forest as 500 and number of neighbours as 8 for k-NN. These settings were also used in evaluation of the ESC dataset, which helps in comparing results.

```
clf = svm.SVC(C=0.1, kernel='linear',random_state=rnd)
clf2 = RandomForestClassifier(n_estimators=500, random_state=rnd)
clf3 = KNeighborsClassifier(n_neighbors=8)
```

After training the classifiers with training set features and corresponding labels, the classifiers were made to predict the classes of the test set's examples from their features. Accuracy was then calculated by comparing the predicted classes with the correct labels. Below is an example of the process for the SVM classifier.

```
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
accuracy = metrics.accuracy_score(y_test,y_pred)
accuracies_svm.append(accuracy)
```

The samples were split randomly five times and the results for classifiers trained with each set are listed in Table 4.1.

Table 4.1 *Prediction results for SVM, random forest and k-NN classifiers.*

Classifier	**1st**	**2nd**	**3rd**	**4th**	**5th**	**Avg.**	**Std.**
SVM	82.2 %	73.3 %	84.4 %	75.5 %	84.4 %	80.0 %	4.7 %
Random forest	80.0 %	82.2 %	82.2 %	86.7 %	73.3 %	80.9 %	4.4 %
k-NN	60.0 %	68.9 %	60.0 %	66.7 %	68.9 %	64.9 %	4.1 %

According to the results, the average accuracies for SVM and random forest classifiers were in the wanted range of 80–90 %, while the k-NN classifier's results were somewhat lower. Still, when compared to a random classifier's accuracy of 33 % all the classifiers' performances can be considered quite successful.

Reported accuracies for SVM, random forest and k-NN classifiers for the ESC-10 dataset were 67.5 %, 72.7 % and 66.7 % respectively. The evaluation setup with ESC-10 was identical to the one used here with the only difference of 10 classes

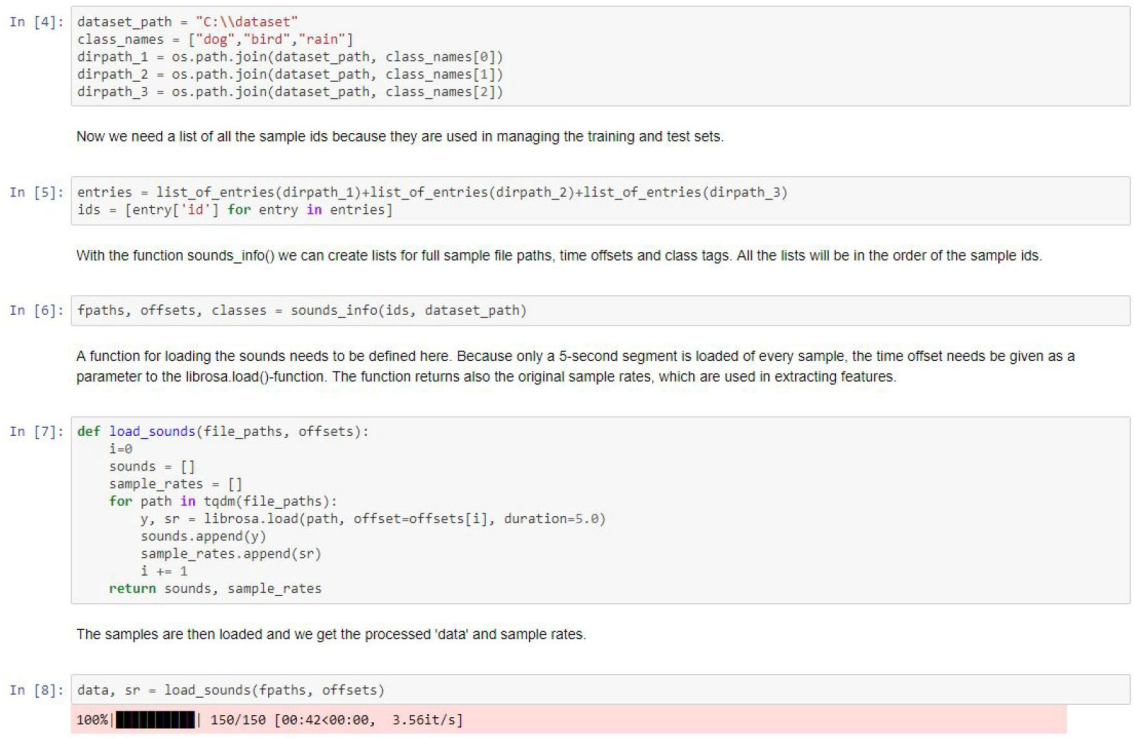

Figure 4.1 *A screenshot of the tutorial Jupyter notebook.*

instead of three and zero-crossing rates as an extra feature beside the MFCCs. [15] Obviously, the classification problem is more difficult when there are more possible classes, and therefore the prediction accuracies are also expected to be lower. The accuracies for the three classifiers trained with ESC-50 dataset containing 50 classes were only 39.6 %, 44.3 % and 32.2 % [15].

Classifier accuracies for UrbanSound8K dataset consisting of 10 classes were approximately 65 % for SVM (with radial basis function kernel) and random forest classifiers, and 55 % for k-NN classifier initialized with 5 neighbors. The features used were MFCCs and the number of cross-validation folds was 10. [19]

Comparing the results of evaluation to the values of ESC and UrbanSound, it can be concluded that the built dataset has potential in testing classifiers. In addition, there was no parameter optimization involved in the evaluation, which might have been the case with the other datasets. The method for building the dataset can therefore be considered suitable for further expanding the number of classes in the dataset. For a more detailed tutorial for using the evaluation code see the Jupyter notebook provided with the dataset (Fig 4.1).

5. CONCLUSIONS

A dataset, named as DBR dataset, was created from samples collected from Freesound. A Python script was written for searching and retrieving samples from Freesound through Freesound API. The created dataset has three classes, 50 samples per class, and the classes are 'dog', 'bird', and 'rain'. Metadata (id, name, description, duration, samplerate, username and license) was saved in yaml format and annotations for five-second segments (starting and ending times and class tags) in text files for each sample. A machine learning application was written for Python, and SVM, random forest and k-NN classifiers were evaluated using the dataset. A Jupyter notebook of the evaluation script was made as a tutorial for using the dataset. The dataset was published in Zenodo [6], and the scripts and the notebook are available at GitHub [1]. The licenses of some of the samples however forbid commercial use.

Building an environmental audio dataset was quite straightforward using the available tools, especially the Freesound client libraries. However, the building process involved plenty of manual work consisting of trying different key words when choosing classes and searching sounds, listening to samples and labeling them. Furthermore, collecting the required 50 samples per class was somewhat challenging mainly due to misleading metadata of samples. Choosing common and consistent sounds, like barking of dogs, was an effective way to relieve the process. Unfortunately this is usually not the case, because solving a particular classification problem starts with an existing set of classes needed to recognize. More attention needs to be given then to the formation of the problem and the selection of search parameters.

Further development ideas for the dataset include publishing only the five-second segments without the original wav files. The pre-extracted features could be distributed, and multiple training and test set splits with corresponding id lists could be prepared. A subset of samples suitable for commercial use could also be published.

With the Jupyter notebook tutorial, using the built dataset should be easy even for people unfamiliar with audio processing. Moreover, the code written for searching and retrieving samples from Freesound will assist in future dataset building.

[1] `https://github.com/vvekl/dbr-dataset`

BIBLIOGRAPHY

[1] AudioSet, webpage. Available (accessed November 3): `https://research.google.com/audioset/dataset/index.html`.

[2] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[3] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[5] W. Dubitzky, M. Granzow, and D. P. Berrar. *Fundamentals of data mining in genomics and proteomics*. Springer Science & Business Media, 2007.

[6] V.-V. Eklund. DBR dataset. Available: `https://doi.org/10.5281/zenodo.1069747`, Dec. 2017.

[7] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of human genetics*, 7(2):179–188, 1936.

[8] F. Font, G. Roma, and X. Serra. Freesound technical demo. In *ACM International Conference on Multimedia (MM'13)*, pages 411–412, Barcelona, Spain, 21/10/2013 2013. ACM, ACM.

[9] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

[10] T. K. Ho. Random decision forests. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 278–282. IEEE, 1995.

[11] J. Janer, S. Kersten, M. Schirosa, and G. Roma. An online platform for interactive soundscapes with user-contributed content. In *AES 41st International Conference on Audio for Games*, 2011.

[12] B. Mechtley, A. Spanias, and P. Cook. Shortest path techniques for annotation and retrieval of environmental sounds. In *ISMIR*, pages 541–546, 2012.

[13] P. Mermelstein. Distance measures for speech recognition, psychological and instrumental. *Pattern recognition and artificial intelligence*, 116:374–388, 1976.

[14] A. Mesaros, T. Heittola, and T. Virtanen. TUT sound events 2017, development dataset. Available: `https://doi.org/10.5281/zenodo.814831`, Mar. 2017.

[15] K. J. Piczak. Esc: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM International Conference on Multimedia*, MM '15, pages 1015–1018, New York, NY, USA, 2015. ACM.

[16] F. Ringeval, A. Sonderegger, J. Sauer, and D. Lalanne. Introducing the recola multimodal corpus of remote collaborative and affective interactions. In *Automatic Face and Gesture Recognition (FG), 2013 10th IEEE International Conference and Workshops on*, pages 1–8. IEEE, 2013.

[17] G. Roma, P. Herrera, and X. Serra. Freesound radio: supporting music creation by exploration of a sound database. In *Workshop on Computational Creativity Support (CHI2009)(accepted)*, 2009.

[18] G. Roma, P. Herrera, M. Zanin, S. L. Toral, F. Font, and X. Serra. Small world networks and creativity in audio clip sharing. *International Journal of Social Network Mining*, 1(1):112–127, 2012.

[19] J. Salamon, C. Jacoby, and J. P. Bello. A dataset and taxonomy for urban sound research. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 1041–1044. ACM, 2014.

[20] S. S. Stevens, J. Volkmann, and E. B. Newman. A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 8(3):185–190, 1937.

[21] T. Virtanen, M. D. Plumbley, and D. Ellis. *Computational Analysis of Sound Scenes and Events*. Springer International Publishing, 2018.