Environmental Sound Classification using Machine Learning

Stefan T. Nastasie (s.nastasie@tilburguniversity.edu), u678169, group 22
Tilburg University, Warandelaan 2, 5037 AB Tilburg, Netherlands

Abstract

Inspired by Dan Stowell, and Mark D. Plumbley and their work on 'Freefield 1010', our team has attempted to create a binary classification experiment in which multiple approaches were used to tell apart city and nature sounds. I describe how the data was pre-processed and transformed, as well as the different algorithms that were used. Furthermore, I include graphs, showing the accuracy of the methods, and results and provide a conclusion to the experiment.

Keywords: Machine Learning; Sound classification; Neural Network; Audio Feature Extraction;

1. Introduction

Image recognition comes to mind when we think about technologies that can identify trends and patterns and classify them. However, the capacity to hear and analyze sounds is another sensory capability that people employ just as much, and in the context of sound recognition for machines, this topic is critically understudied compared to image recognition.

Learning from previous research (Cowling & Sitte, 2003), our group understood that models and approaches used in speech and voice recognition might prove to be unreliable. Instead, we decided to use an Artificial Neural Network approach, a method with proven results in recognizing urban sounds (da Silva et al. 2019), and K - nearest neighbors (k-NN), which was first proposed and tested by Bountourakis, et al. in 2015.

Despite the fact that a potential application for these sound classification algorithms, mostly in embedded systems, has been discussed in previous works, our team has made an effort to simplify the process of extracting audio features and implementing a simple, yet possibly effective model. For the scope of our task, the goal was to apply previously gained knowledge to the recognition of soundscapes, and by gaining insight into the obtained results from previous accomplishments in this field, we understood that there is much practical interest and potential related to Environmental Sound Recognition (ESR). It should be noted that due to the vast amount of categories of environmental sounds that can occur naturally or artificially, we limited our experiment to a binary classification approach.

2. Methods

The dataset that was used is the one mentioned earlier, 'freefield1010', put together by Dan Stowell and Mark D. Plumbley. It originally consisted of 17807 sound recordings with a length of 10 seconds. The dataset came in the form of .wav files, together with their respective JSON file, which included information about the sound file, such as the author, tags, etc. To be able to select the files we needed out of the whole dataset which was contained in 10 separate folders, we created a python script that looped through the entire collection of JSON files and only selected the ones that contained the tag 'city' (562 recordings) or 'nature' (905 recordings). We then created a new folder that included all needed sound files and JSON files.

To transform the sound files into numerical data that could be interpreted by our models, we used the python module 'Librosa', a package used for sound and music analysis. Since our dataset consists of single-channel sound files, we used a method called 'Mel-frequency cepstral coefficients'. We stored the output into NumPy arrays, which we saved for later use and easier access together with their respective labels as 0 - city and 1 - nature.

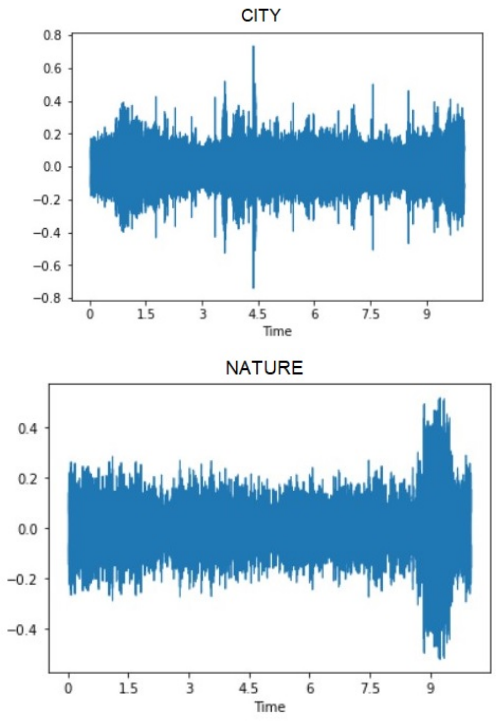

Figure 1: Waveforms of city and nature sounds

We wanted to remain with the methodologies that produced the greatest results in the past, thus we picked two distinct Neural Network architectures and a simple k-nearest Neighbors algorithm to develop our models. We generated our k-NN model with the help of the prebuilt k-NN algorithm in the 'sci-kit learn' package and cycled through an array with the numbers [1,3,5,7,9,11,13,15, 17]. The number of neighbors was passed as a parameter using this array. A basic Multilayer Perceptron model was also available in the 'sci-kit learn' package, which we utilized with two different opti-

mizers ('adam', 'lbfgs'), the 'relu' activation function and a list of hidden layers that comprised 10, 15, 30, 50, and 100. The 'Sequential' model from the Tensorflow(Keras) package was used for a more complex Convolutional Neural Network (CNN) approach. We chose this model since our intended outcome was a simple output (a 1 or a 0). The model consists of four dense hidden layers that use a 'relu' activation function and an 'adam' optimizer.

3. Results

This section's results are divided into three sections, each detailing a model's outcome. It should be mentioned that the data was split into train and test data in a 4:1 ratio using 10-fold cross-validation for each model. A validation set was included in both the Sequential model and the MLP Classifier, on which our team did hyper-parameter tuning. The displayed results represent the mean extracted from all the cross-validation folds.

3.1 KNN

As previously stated, the only passable argument for the k-Nearest Neighbors model was the number of neighbors itself. The worst test-set accuracy (69%) was shown by the model that received 1 neighbor as a parameter while displaying the best train set accuracy(99%). This is to be expected since the model tends to overfit the lower the number of neighbors is. As shown in (Fig. 2), the test set accuracy gradually increased until it peaked at 76% with 17 neighbors, while the training set began to decline while remaining at around an 80% mark. We terminated parameter testing at 17 neighbors because the test set accuracy did not improve after this peak.

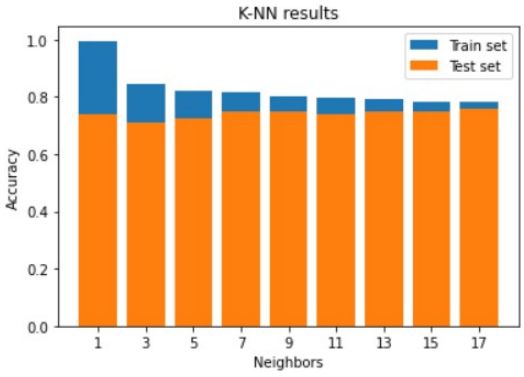

Figure 2: Results for the k-NN model

3.2 MLP

We decided not to present two of the three parameters in the Multilayer Perceptron model (optimizer and activation function) since they had little to no influence on the findings. The number of hidden layers passed while constructing the model was the only notable difference. Test set accuracy fluctuates between 70 and 76 percent (Fig. 3), whereas train set accuracy grew steadily from 78 percent to 89 percent. The 100-layer model had the best accuracy on the test set, and it also had the best accuracy for the train test. This is to be expected because this model underfits as the number of hidden layers is decreased. A higher number of hidden layers did not seem to affect the accuracy on the test set, therefore we stopped at 100.

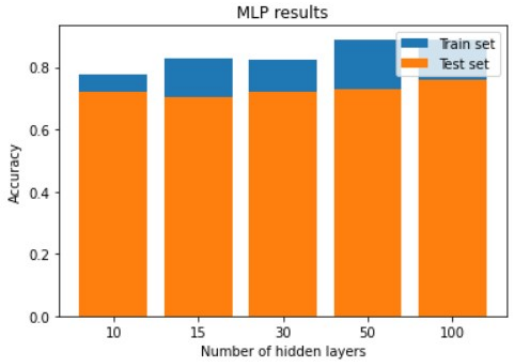

Figure 3: Results for the MLP model

3.3 CNN

Changing the optimizer and activation function for our Convolutional Neural Network had a negligible effect on prediction accuracy, similar to the MLP. The dropout rate, which is responsible for randomly deactivating neurons and their connections in order to prevent the network from depending on only a few nodes, was the sole parameter we modified during the iterations. The dropout rates on all hidden layers at the same time ranged from 0.1 to 0.6, increasing by 0.1 per iteration, while the input layers remained at 0.8 throughout. We can see that lower dropout rates resulted in the highest accuracies on both the test set and training set, with 0.2 being the top score (74%) for the test set, while larger dropout rates result in lower accuracies (Fig. 4). We noticed a sharp decline in accuracy from 0.5 to 0.6, which is why we decided to stop iterating at that point.

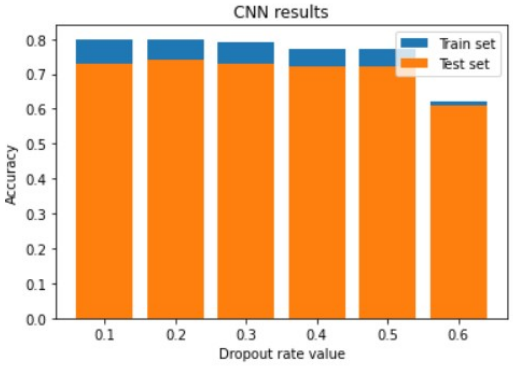

Figure 4: Results for the CNN model

Table 1: Comparison of models

Model	Accuracy
k-NN	76%
MLP	76%
CNN	74%

4. Discussion

The most time-consuming element of the experiment, according to our assessments, was feature extraction and file sorting. On a typical i5 Intel 10th Generation CPU, the entire operation can take several minutes. On bigger datasets, this may become an issue, however, this clearly depends on the computer's quality. We acquired industry-standard results based on our findings, which typically range between 70 and 90 percent. The CNN classifier was the longest to compile, taking roughly a minute, whereas the k-NN and MLP classifiers just took a few seconds.

A challenging aspect of this research assignment was downloading the actual dataset, mainly due to the unresponsiveness of the servers, therefore, torrenting the files proved to be a better solution. Another difficult task was converting the sound files into numerical data because the functions we used from the Librosa package returned 3-dimensional arrays, when all the algorithms we used required 2-dimensional inputs. As a result, we had to use the mean to alter and reshape one of the features.

Although the results of our versions of these popular classification models do not match the accuracy of earlier results in other studies, they do show the crucial factors to consider when creating classification algorithms to address this unique situation. The significance of producing samples with a large context, the volume of training data, and the lack of in-depth expertise about the topic are the main reasons behind this. I believe that a larger number of samples and a more extensive analysis of the subject will be required in future studies.

5. Conclusion

In this paper, I attempted to demonstrate our team's efforts to create our own binary classification models using popular sound classification approaches. We carefully selected two different categories of environmental sounds (city and nature) and converted the sound files into numerical data that could be evaluated and used to train our algorithms. As could be seen in the results section, all of the models produced similar results, with the MLP and k-NN having slightly better results.

Our results did not perfectly match those of prior studies, but they can be improved with a more in-depth analysis of the field and the exploration of other classification algorithms. Finally, I feel that the mostly untapped topic of Environmental Sound Recognition has the potential to be useful in a variety of aspects of our society, and I hope that our research will serve as a catalyst for future research.

References

Bountourakis, V., Vrysis, L., & Papanikolaou, G. (2015). Machine learning algorithms for environmental sound recognition. *Proceedings of the Audio Mostly 2015 on Interaction With Sound - AM '15*. doi: 10.1145/2814895.2814905

Cowling, M., & Sitte, R. (2003). Comparison of techniques for environmental sound recognition. *Pattern Recognition Letters*, *24*(15), 2895–2907. doi: 10.1016/s0167-8655(03)00147-8

Nolasco, I., & Benetos, E. (2018). To bee or not to bee: Investigating machine learning approaches for beehive sound recognition. *NASA/ADS*.

Silva, B. d., Happi, A. W., Braeken, A., & Touhafi, A. (2019). Evaluation of classical machine learning techniques towards urban sound recognitionon embedded systems. *Applied Sciences*, *9*(18), 3885. doi: 10.3390/app9183885

Stowell, D., & Plumbley, M. (2014, Jan). An open dataset for research on audio field recording archives: Freefield1010. *" An Open Dataset for Research on Audio Field Recording Archives: freefield1010*.