Audio Intent Detection Classification Problem

Alessandro Masala
DAUIN
Politecnico di Torino
Torino, Italy
s305206@studenti.polito.it

Alessandro Turrin
DAUIN
Politecnico di Torino
Torino, Italy
s317721@studenti.polito.it

***Abstract*—In this report we introduce a possible approach to the *Audio Intent Detection* classification problem. Furthermore, the proposed approach consists in extracting the spectrogram of each audio signal and split it into a specific number of chunks. For each chunk, various summary statistics are computed and used as input for two classification models.**
The produced code is available at the following link:
Audio Intent Detection - Classification Problem

I. Problem Overview

The task of *intent detection* is crucial in the field of Natural Language Processing (NLP) [1] and speech recognition. This task is particularly important in the development of voice assistants and other audio-based applications, as it allows the system to understand and respond appropriately to the user's requests and commands.
The goal, given an input audio sample, is to predict both the *action* requested and the *object* that is affected by the action. The task has been fulfilled using a provided dataset that consists in a collection of audio file in WAV format [2]. It is divided in two parts:

- **development** set: containing 9,854 recordings, each one with *action* and *object*
- **evaluation** set: comprised of 1,455 recordings

We will use the development set to build a classification model to correctly label the recordings in the evaluation set. We can make some considerations based on the development set.
Firstly, the problem is not perfectly balanced. In fact, we can see in Figure 1 that the number of recordings belonging to the classes is not the same for all classes. Hence, it will be necessary to equalize the number of recordings for each class. Secondly, although all signals have been sampled at 22050Hz, recordings duration differs. Figure 2 shows how such recordings durations are distributed. There are some outliers that significantly differs from the mean duration (2.64 seconds). Thanks to a manual inspection, we can affirm that some signals contains silence following the utterance, while some other signals contains indistinguishable sounds following the utterance, therefore this noise might represent a problem in the task of recognize the instruction. Since the entries has varying length, we needed to find some approach to extract the same number of features from all the data points.

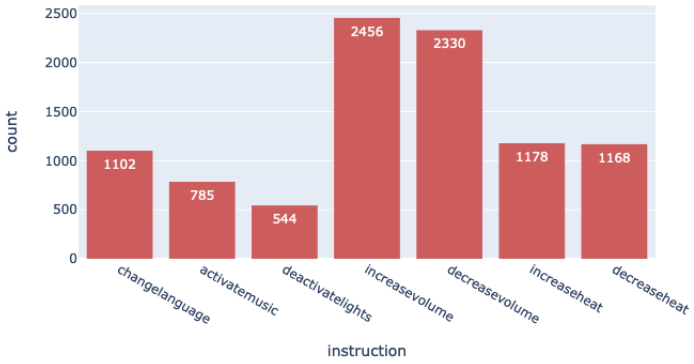

Fig. 1. Number of recordings per class

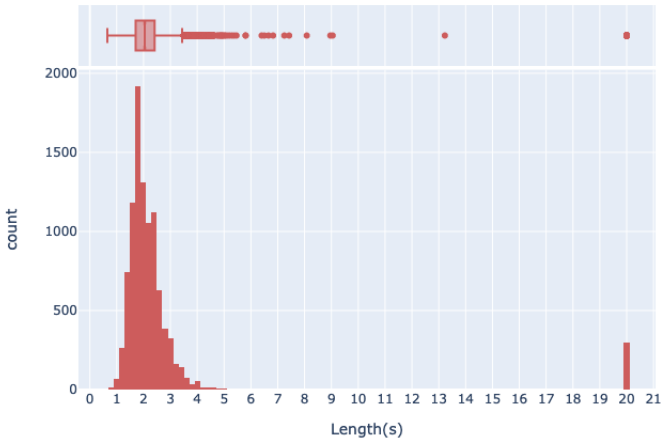

Fig. 2. Distribution of the duration of the recordings

II. Proposed Approach

A. Data Preprocessing

1) Dataset cleaning: the dataset was made of various features that were useless for the aim of the project. Therefore, we decided to remove those features in order to keep only the most relevant ones.

2) Recordings preprocessing: in order to have a better understanding of the signals we were analyzing, we decided to carry out a visual inspection of the recordings, both in time and frequency domain. Hence, we noticed that some audio signals can contain some pause before or/and after the utterance. An example is shown in Figure 3.

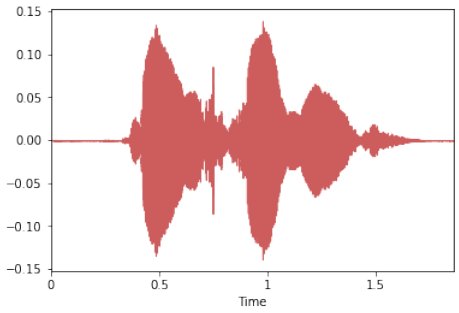

Fig. 3. Representation of a recording in time domain

Therefore, we decided to use the spectrogram of the signals in order to laverage both time and frequency features. Particularly, we used the STFT (Short-Time Fourier Transform) [3] spectrogram and the MFCC (Mel-Frequency Cepstral Coefficients) [4] spectrogram. An example of the spectrograms of a signal is shown in Figure 4.

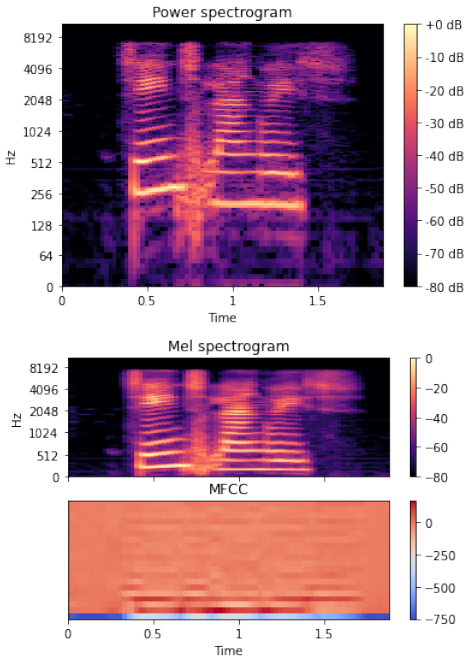

Fig. 4. Example of signal spectrograms. (a) Power spectrogram; (b) MEL spectrogram and related MFCC.

3) The class imbalance problem: the class imbalance problem [5] is a common problem in machine learning. It occurs when the training dataset has an unequal distribution of classes, leading to a potential bias in the training model. There are several methods that can be used to address class imbalance, however we decided to use SMOTE (Synthetic mynority over-sampling technique) [6]. SMOTE is an algorithm that performs data augmentation by creating synthetic data points based on the original points. The advantage of this technique is that we are not generating duplicates, but rather creating synthetic data points that are slightly different from the original data points. Using this technique we were able to balance the recordings for each class, as shown in Figure 5.

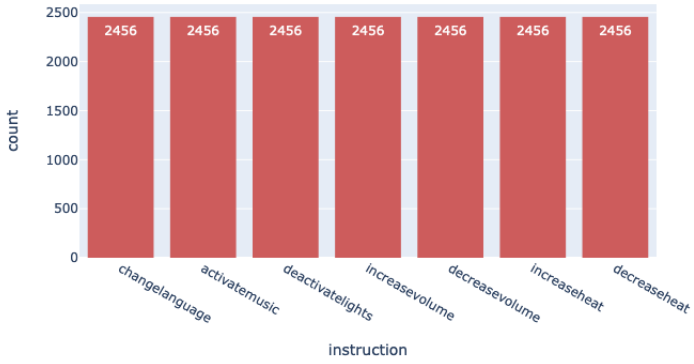

Fig. 5. Number of recordings per class

4) Features engineering: as shown in Figure 2, the recordings duration differs in many cases. Therefore, we decided to exploit the characteristics of the spectrograms in order to find a suitable solution for the problem. Furthermore, a spectrogram can be seen as an *N x M* matrix (N time bins, M frequency bins), we decided to divide it into n_x rows and n_y columns, in order to obtain $n_x \cdot n_y$ blocks. Then, we decided to compute statistics on each block, in our case μ (mean) and σ (standard deviation). This procedure was extremely useful for two main reasons:

- **Reduction of the number of features** taking $N \cdot M$ blocks, we would have the same features produced by the spectrogram. Reducing the number of blocks consequentially reduces the number of features
- **Generation of a uniform number of features** selecting a number that is not a function of *N* and *M*, we can generate the same number of features for all signals

B. Model selection

The following algorithms have been tested:

- **Random Forest**: this algorithm leverages multiple decision trees (trained on various subset of data and features) to make predictions. Although it normally deflects the overfitting problems, it still preserve some degrees of interpretability. Its preformances scale with the number of estimators until a certain point [7]. Random forests work on one feature at a time, hence normalization is not required. We decided to use Random Forest because it shows nice performances on audio signal classification problems [8].

- **MLP**: the Multilayer Perceptron (MLP) model is the simplest form of DNN suited to classification tasks, and it provides decent off-the-shelf performance in audio detection problems [9]. We decided to use MLP based on the fact that classic machine learning models have distinct advantages to deep neural networks in many tasks, however they hardly can match the performance of even the simplest deep neural network in the task of audio classification. Furthermore, MLP can be precisely tuned to be accurate and relatively quick to train.

C. Hyperparameters tuning

There are two main sets of parameters to be tuned:

- n_x and n_y for the preprocessing
- *Random Forest* and *MLP* parameters

TABLE I
HYPERPARAMETERS TUNING

Model	Parameters	Values
Preprocessing	n_x, n_y	$2 \rightarrow 40$, step 2
Random Forest	n_estimators	[300, 500, 1000, 1200, 1500, 2000]
	max_depth	[None, 2, 5, 10, 50]
	criterion	[gini, entropy]
MLP	max_iterations	[10000, 100000, 1000000]
	learning_rate	[constant, invscaling, adaptive]

In order to find the best combination of n_x, n_y we trained the default configurations of parameters for *Random Forest* and *MLP* and evaluate their performances on the accuracy score. After choosing the value for n_x, n_y we decided to run a grid search both on Random Forest and MLP, based on hyperparameters shown in Table I. Moreover, the grid search has been performed using, one at a time, the STFT features and the MFCC features.

III. RESULTS

During the tuning of n_x, n_y we noticed that Random Forest performed better than MLP. Both classifiers achieved reasonable results for $n_x = n_y = 32$. Using this values we ran the grid search for the classifiers using an 80/20 train/test split.
The best configuration for Random Forest was found for $\{$*criterion=gini, max_depth=None, n_estimators=2000*$\}$ using STFT features (accuracy = 0.819). On the other hand, he best configuration for MLP was found for $\{$*learning_rate=adaptive, max_iter=1000000*$\}$ using STFT features (accuracy = 0.752).
We trained the best performing Random Forest and MLP on all available development data. Then the best model has been used to label the evaluation set. The best public score obtained is 0.560 obtained with Random Forest. Table II shows the results of our trainings and the best result obtained in the evaluation.

IV. DISCUSSION

The proposed approach obtains results that outperform the naive baseline, exploiting both time and frequency freatures.
We have empirically shown that the selected classifiers performs differently. Particularly, considering the tests carried out on the same spectrogram, Random Forest performs better than MLP.
There are some aspects that might be worth considering to further improve the obtained results:

- Although MFCC and STFT are often used in speech recognition tasks [10], other approaches based on CNN (Convolutional Neural Network) for automated features extraction might be extremely useful
- Even though classifiers can achieve discrete performances, they are not comparable with the performances that can be obtained using a Neural Networks. Therefore, a further analysis of the problem can be done using Neural Network, in order to drastically improve the performances.

In conclusion, the obtained results are promising considering the fact that the datasets available are limited. However, there is the possibility to improve the results.

TABLE II
FINAL RESULTS

Model	Parameters	Spectrogram	Results	
			Train	Test
Baseline				0.334
Random Forest	n_estimators=2000 max_depth=None criterion=gini	MFCC	0.779	
MLP	max_iterations=1000000 learning_rate=adaptive	MFCC	0.683	
Random Forest	n_estimators=2000 max_depth=None criterion=gini	STFT	0.819	**0.560**
MLP	max_iterations=1000000 learning_rate=adaptive	STFT	0.752	

The values highlighted in bold outperform the baseline value

REFERENCES

[1] Aditya Jain, Gandhar Kulkarni, and Vraj Shah. Natural language processing. *International Journal of Computer Sciences and Engineering*, 6:161–167, 01 2018.

[2] Eric W. Fleischman. WAVE and AVI Codec Registries. RFC 2361, June 1998.

[3] Nasser Kehtarnavaz. Chapter 7 - frequency domain processing. In Nasser Kehtarnavaz, editor, *Digital Signal Processing System Design (Second Edition)*, pages 175–196. Academic Press, Burlington, second edition edition, 2008.

[4] Md. Sahidullah and Goutam Saha. Design, analysis and experimental evaluation of block based transformation in mfcc computation for speaker recognition. *Speech Communication*, 54(4):543–565, 2012.

[5] Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intell. Data Anal.*, 6(5):429–449, oct 2002.

[6] Kevin W. Bowyer, Nitesh V. Chawla, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *CoRR*, abs/1106.1813, 2011.

[7] Thais Oshiro, Pedro Perez, and José Baranauskas. How many trees in a random forest? volume 7376, 07 2012.

[8] Fatemeh Saki, Abhishek Sehgal, Issa M. S. Panahi, and Nasser Kehtarnavaz. Smartphone-based real-time classification of noise signals using subband features and random forest classifier. *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2204–2208, 2016.

[9] Ilia Zenkov. sklearn-audio-classification. https://github.com/IliaZenkov/sklearn-audio-classification, 2020.

[10] Todor Ganchev, Nikos Fakotakis, and Kokkinakis George. Comparative evaluation of various mfcc implementations on the speaker verification task. *Proceedings of the SPECOM*, 1, 01 2005.