Audio Intent Detection Problem

Faezeh Saeedian
Politecnico di Torino
s301308
s301308@studenti.poltio.it

Reza Barati
Politecnico di Torino
s301309
s301309@studenti.poltio.it

***Abstract*—In this report, two potential approaches to the intent detection problem for an audio data set are presented. Two different approaches were employed after extracting the MFCC from each audio signal. The first approach involves dividing the MFCC into a specified number of blocks, and computing various statistics for each of them, that are used as inputs for a classification model(in this case XGBoost). The second approach is, using the MFCC as the input for a CNN model. Both methods outperform a simple benchmark established for the problem, and the CNN model shows superior performance in comparison to XGBoost.**

***Index Terms*—intent detection, MFCC, XGBoost, Convolutional Neural Network (CNN),**

I. Problem overview

Intent detection refers to the task of determining the purpose behind a user's natural language expression, such as a spoken or written sentence.

This project focuses on identifying the intent in an audio data-set consisting of recordings of multiple speakers, with accompanying information about their age range, gender, first language spoken, main language spoken during daily activities and fluency level. The objective is to predict both the action that demonstrates the type of action required through the intent and the object that demonstrates the device involved by intent. In this project we combined these two columns as a string to reach a single one for each audio. We use this new column as our target.

The data-set is split into two parts:

- A development set with 9855 recordings for which a label has been assigned.
- An evaluation set with 1455 recordings.

We will utilize the development set to develop a classification model for correctly categorizing the points in the evaluation set.

When exploring the development set, there are some points. The data-set is almost imbalanced, the number of audios in the "increase volume" and "decrease volume" classes being around 2500, while for the rest of the classes, it is approximately half that amount, that is shown in Fig. 1. Additionally, the sample width and number of channels for all audios are 8 bits and 2 (stereo), respectively, and all recordings were sampled at a frequency of 22.05 kHz.

The length of the audio records are different, with duration rang from 1 to 5 seconds, and some are deviant significantly from the average duration that are considered as outliers. The distribution of audio duration is shown in Fig. 2. An inspection of the signals reveals that some of them include silent or noisy segments at the end that do not contain important information. The records have different lengths which means the amount of features which is extracted from each record varies, while the majority of classification models require a fixed number of features. Therefore, we had to find a way to convert audios to the same length before extracting features of them.

Identify applicable funding agency here. If none, delete this.

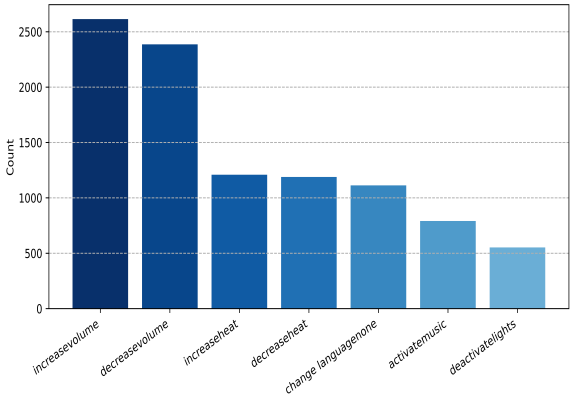

Fig. 1. Frequency of labels.

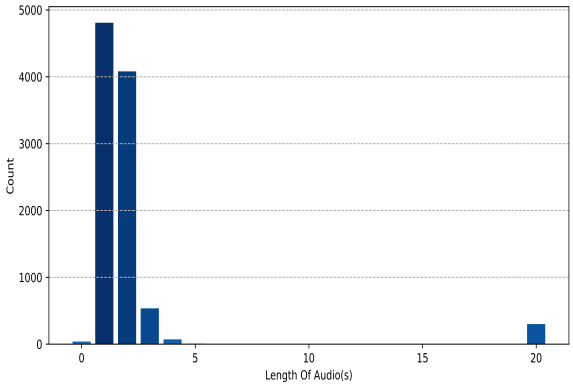

Fig. 2. Distribution of audios duration.

Audio in the time and frequency domains can be visually

analyzed to give us a better understanding of the kind of data we are working with. These two domains provide different and important information for each signal. A single time-domain audio is shown in Fig. 3.

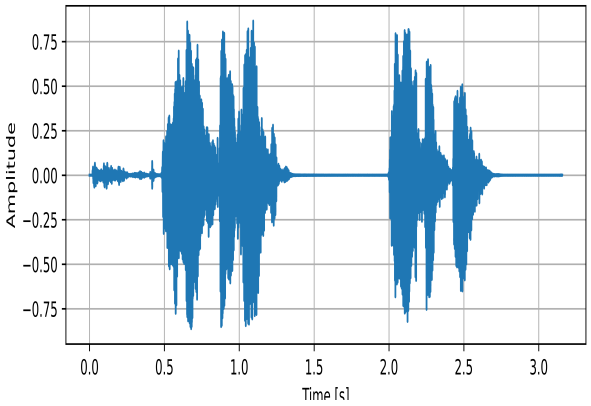

Fig. 3. Representation of an audio in the time domain.

The signal in the frequency domain is shown on a decibel scale in Fig. 4, indicating varying energy levels among frequencies. The decibel scale is logarithmic and provides a wider representation of important frequencies and is used due to its similarity to human perception [1].

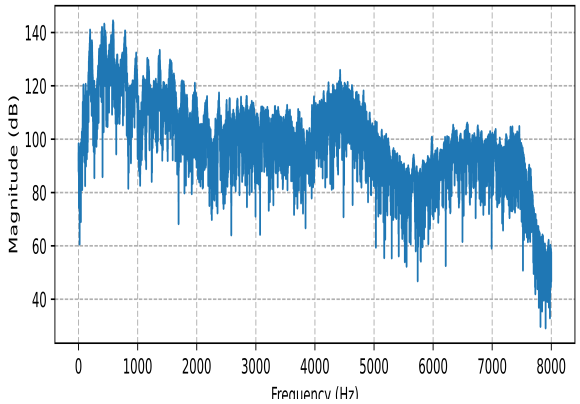

Fig. 4. Magnitude of an audio.

II. Proposed Approach

A. Preprocessing

The audio in the data-set has varying lengths, which is a problem when we want to use them as inputs for classification that require a consistent length. To resolve this issue, we keep the first 5 seconds of audio which their duration is longer than 5 seconds, because rest of that do not have helpful information, while audios that are shorter than 5 seconds, are extended to 5 seconds by padding. This ensures that all the audio samples have the same length, making them suitable to be used as inputs for the algorithms.

We have come to understand that the audio files contain significant information in both the time and frequency domains. Both of these can be used when using the MFCC of each audio. MFCC (Mel-Frequency Cepstral Coefficients) are computed from the power spectrum of an audio signal and represent the shape of the spectrum in terms of a set of coefficients. MFCC capture important aspects of the audio signal, such as the spectral envelope, which helps to capture the perceived quality of an audio [2]. Fig. 5 shows an example of a MFCC of an audio.

One approach to analyzing a MFCC is to divide it into smaller blocks in the time axis, and then compute some statistics such as the mean, standard deviation, min, max and etc for each block that can provide more detailed understanding of the content of the audio signal and get a one-dimensional vector for machine learning algorithms. Another way to analyze MFCC is to use the 2D array as input for a Convolutional Neural Network (CNN).

After extracting features of audios by using MFCC, we should prepare our data-set for a machine learning algorithm by converting the categorical columns to numerical values. This will allow the algorithm to effectively process and make predictions based on the data. We have two types of categorical data in our data-set, nominal and ordinal. Nominal data consists of columns where there is no inherent order between the labels (in this case, first language spoken, current language and gender), while ordinal data(age range and fluency level) consists of columns that the order of the labels is important. Therefore, we use OneHotEncoder for the first type and OrdinalEncoder for the second one. Also for categorical labels, we use LabelEncoder to gain numerical values. at the end, the obtained numerical values and extracted features of MFCC are merged to be ready as an input of XGBoost classifier.

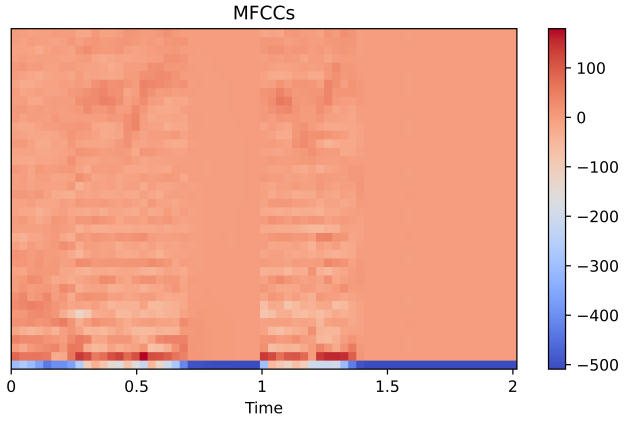

Fig. 5. MFCC of an audio.

B. Model selection

The following algorithms are evaluated:

- XGBoost (Extreme Gradient Boosting): this model has been shown to work well on audio signals [3]. This

algorithm is a powerful and efficient machine learning algorithm that is an implementation of gradient boosting. It uses decision trees as the base model. The basic idea behind the algorithm is to repeatedly train decision trees on the residual errors of the previous tree, so that each new tree focuses on the mistakes of the previous one. The final ensemble model is a weighted sum of all the individual decision trees. XGBoost also includes a number of techniques to improve the performance, such as regularization, which helps to prevent overfitting, and parallel processing to speed up the training process.

- CNN: convolutional neural network (CNN) is a type of deep learning neural network that is typically used for image analysis, but can also be applied to audio data [4]. In the context of audio data classification, CNN can be used to classify audio by analyzing their MFCC. One of the key advantages of using a CNN for audio classification is that, it is able to learn both local and global features from the input data, which allows it to be robust to variations in the audio data, such as background noise and different recording conditions.
  CNN takes the MFCC of an audio as input and applies a series of convolutional, pooling, and fully-connected layers to extract features from the input data. These features are then used to classify the audio into one of several predefined classes.

The optimal hyperparameter configuration for both classifiers was determined through a grid search process, as described in the next section.

C. Hyperparameters tuning

There are two key groups of hyperparameters to be optimized. One group is related to the MFCC feature extraction process and the other group is linked to the classifiers. We can consider them as separate entities and adjust the parameters for feature extraction first before moving on to those for the classifiers.

The most important parameters in MFCC are ”n-mfcc”, ”n-fft” and ”hop-length”. The term ”n-mfcc” denotes the quantity of MFCC coefficients to calculate. The length of the FFT window is specified by ”n-fft,” and ”hop-length” indicates the number of samples between each frame.

To handle the imbalance data, we employed three strategies. Firstly, we utilized stratified splitting, when split the dataset into train and test, to preserve the proportion of values in the sample as same as the input data. Additionally, we incorporated class weight, which gives higher importance to the minority class and encourages the model to focus more on its samples. This helps balance the impact of each class in the final model. Lastly, we used the F1 Score to evaluate the model's performance, as it is more appropriate for imbalanced data.

We separated the development set into a training portion (80 percent) and a testing portion (20 percent). Then, we performed a grid search on three parameters of MFCC by training both Xgboost and CNN with fixed configurations, and the performance will be evaluated based on the macro f1 score and accuracy score obtained. the two approaches for solving this problem differ significantly, since their architecture are different, so the parameters for MFCC are adjusted independently.

Once the value for MFCC parameters was chosen for each classifier, a grid search was carried out separately on Xgboost and CNN classifiers based on the hyperparameters are shown in Tables I and II.

TABLE I
HYPERPARAMETERS TUNING FOR XGBOOST

Parameter	Values	F1-score	Accuracy
n-fft	[128, **256**,512,1024]	48.7	49.4
hop-length	[64, **128**,256,512]		
max-depth	[4,**5**,6]	49.1	50.3
min-child-weight	[2,**4**,6]		
min-samples-spli	[2,**3**,4]		
sampling-method	[**uniform** , gradient-based]		

[a]Bold numbers represent the best parameters.

TABLE II
HYPERPARAMETERS TUNING FOR CNN

n-fft	hop-lenght	n-mfcc	Opt	Lr	F1-score	Accuracy
2048	1024	[**40**, 50, 60]	Adam	Default(0.001)	78.5	76.1
1024	512	[40, **50**, 60]	Adam	[**0.001**, 0.0001]	84.6	85.7
			SGD	[**0.01**, 0.001]	77.1	75.6
512	248	[50, **60**]	Adam	Default	83.2	81.7

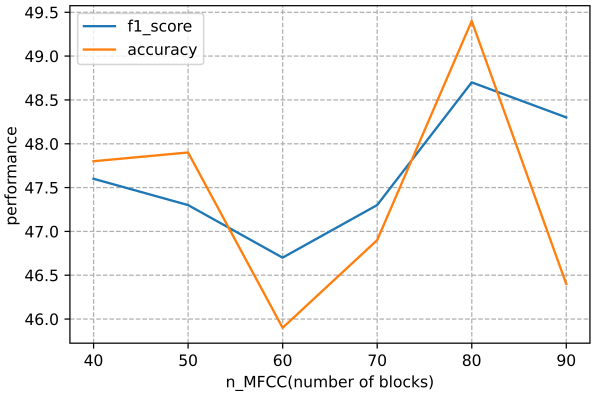

Fig. 6. Performance of Xgboost based on the number of blocks.

III. RESULTS

The process of finding the optimal values for the number of blocks (n-mfcc) in training Xgboost is displayed in Figures 6 and the number of epochs for training CNN is shown in 7. The results show that a value of 80 for the number of blocks, also, 50 for the number of epochs produce the best results since, the performance of the CNN does not get better and stays constant. Additionally, Tables I and II demonstrate that the optimal MFCC parameters for Xgboost were determined to be [n-fft = 256 and hop-length = 128], while the best configuration for

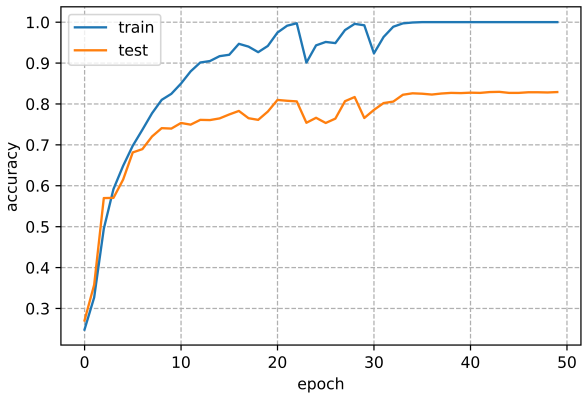

Fig. 7. Performance of CNN based on the number of epocs.

CNN was found to be [n-fft = 1024, hop-length = 512, and n-mfcc = 50].

The optimal configurations for Xgboost was determined as [maximum depth=5, minimum child weight= 4, minimum samples split= 3, sampling technique= uniform] (F1 score = 0.491, accuracy score = 0.503), while the best configuration for the CNN was established as [optimizer= Adam, learning rate= 0.001] (F1 score = 0.846, accuracy score = 0.857). Both classifiers deliver satisfactory results.

After training the most effective Xgboost and CNN models on the entire development dataset, we used these models to classify the evaluation set. The obtained score for the Xgboost model was 0.460 and for the CNN model was 0.857.

IV. Discussion

The proposed approaches obtain results that outperform the naive baseline defined(33.4 percent), By utilizing both time and frequency based features. Through experimental evidence, it has been demonstrated that the CNN classifier is more effective for this particular task in terms of accuracy and f1-score compared to XGBoost. The following are some factors that might be taken into account to enhance the outcome more:

- Use pre-trained models for extracting features, like VGGish [5], which have been trained on a large dataset, are used to extract meaningful features from audio data. Without starting from scratch and training a model, these feature extractors can be used to extract features from audio recordings. This makes feature extraction faster and more precise.
- Explore more in grid search. Only a few hyperparameters have now been studied. An even better setup can increase the results. Moreover, implementing other classification models may lead to better results.

References

[1] Goldstein, E. B. (1989). Sensation and perception (3rd ed.). Wadsworth/Thomson Learning, 1989.

[2] Mandel, M.I. and Ellis, D.P., 2005. Song-level features and support vector machines for music classification.

[3] Dahiya, A., 2019. Audio instruments identification using CNN and XGBoost (Doctoral dissertation, Dublin, National College of Ireland).

[4] Hershey, S., Chaudhuri, S., Ellis, D.P., Gemmeke, J.F., Jansen, A., Moore, R.C., Plakal, M., Platt, D., Saurous, R.A., Seybold, B. and Slaney, M., 2017, March. CNN architectures for large-scale audio classification. In 2017 ieee international conference on acoustics, speech and signal processing (icassp) (pp. 131-135). IEEE.

[5] Moura, S. and Queiroz, M., 2018. Singing voice detection using vggish embeddings. In 19th International Society for Music Information Retrieval Conference, Paris, France.