Intent detection of audio classification (NLP)

Hesam Khanjani Kakroodi
Politecnico di Torino
Student id: s310141
s310141@studenti.polito.it

Parsa Taati
Politecnico di Torino
Student id: s309760
s309760@studenti.polito.it

***Abstract*—In this report, we are going to investigate the sound processing for the purpose of intent detection. Intent detection is crucial for NLP and speech recognition systems to understand the purpose of a statement. The suggested method involves taking the spectrogram and Mel-spectrogram of each audio signal, Implementation pre-processing and using them as input for a CNN classification model. Convolutional Neural Networks (CNNs) are effective in intent detection tasks as they can extract relevant features, handle large amounts of data, and perform classification.**

I. Problem overview

Intent detection is a problem in the field of natural language processing (NLP) and speech recognition where the goal is to identify the underlying intention or purpose of a spoken or written statement. In the case of the problem we face with multi class and multi label dataset you described, The objective of this project is to determine both the action being requested and the object that is impacted by the action, based on the input audio sample provided [1].

The dataset is divided into two parts:

• A development set, containing 9853 recordings for which we want to train our prediction model based on them

• An evaluation set, comprised of 1454 recordings for which we could use them as a test set for calculating the accuracy, after training our model.

The first stage involves pre-processing the data to extract relevant features, which can help improve the accuracy of the analysis and the second step involves choosing the training dataset to implement the classifiers. The database that we use in the project include attributes such as id, path, speakerId, action, object, Self-reported fluency level, First language, Current language, gender, age range. Each record is characterized by several attributes. The following is as a short description of some species of them that we want to use during the project:

• ID: a numerical identifier of the audio
• Path: audio address
• Action: the action that speakers want to do
• Object: the object that we want to action on it

For training and validation of our model, we split the development set to 80% and 20% respectively.

II. Proposed approach

A. Preprocessing

First of all, data are loaded, and the raw data must be pre-processed to be transformed into a format that can be easily analyzed for further use. We convert our audios to numerical format that we can process it and then convert it to spectrogram (feature extraction). The dataset consists of audios with different duration and Some of them have noise and silent areas. To reduce errors, we use a trim method. The trim function is used to remove silent portions or leading/trailing silence from an audio signal. The function accepts the audio signal as an input and returns the trimmed audio with the silent parts removed. This can be useful for removing unwanted background noise and ensuring that the audio signal only contains the desired content. In figure 1 and figure 2 we can see the effect of trimming on sample audio also in figure 3 we can see that at 10 db reach the best result (the sampled audio is 20 second and the main section of speaking is 2 second).

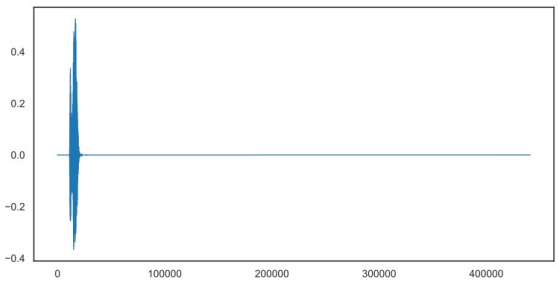

Fig. 1. Raw Audio Before Trimming

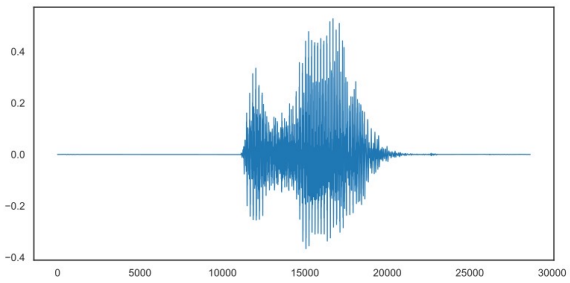

Fig. 2. Raw Audio After Trim in 60 db

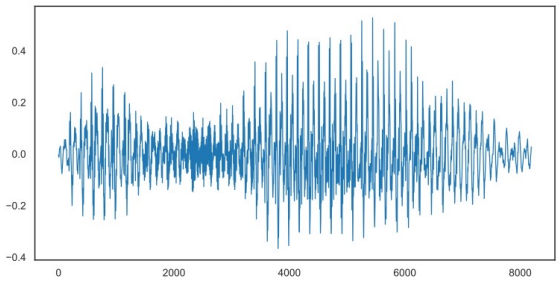

Fig. 3. Raw Audio After Trim in 10 db

In this project, we have decided to normalize the data by using the Z-Score method. Z-Score normalization is a technique for feature normalization in a CNN audio classification where the audio features are transformed such that their distribution has a mean of zero and a standard deviation of one. The formula for Z-score normalization is:

$$X\_norm = (X - mean(X))/std(X) \quad (1)$$

Where X is the original feature and X-norm is the normalized feature. By transforming the features using Z-score normalization, the features are centered around the mean and scaled based on the standard deviation. By normalizing the data, we aim to balance the significance of each feature, avoiding the domination of a single feature which could result in a faster convergence and better performance in the task of audio classification [2].

In Figure 4 and 5 we can see the frequency on the left side and over time how they are changing now we want to get our Mel spectrogram (melodic) because we want to express the frequency of the audio that we hear usually and now the frequency that we can hear is little more accentuated.

Mel-frequency Cepstral coefficients (MFCCs) are a set of coefficients that make up the Mel-frequency cepstrum, a representation of the audio that is based on a Cepstral representation (a "spectrum-of-a-spectrum"). The Mel-frequency cepstrum differs from a regular cepstrum in that the frequency bands are spaced equally on the Mel scale, which better mimics the human auditory system's response compared to the linear spacing used in a regular spectrum. As a result, MFCCs are often used in speech and music processing tasks such as speech recognition, speaker identification, and genre classification [3].

Padding on audio MFCCs refers to adding zero values to the end of an audio clip in order to make it have a fixed length. This is typically done when working with sequences of varying lengths, to ensure that they can be processed consistently by machine learning algorithms.

The benefits of padding audio MFCCs are:

• Consistent input length: Padding ensures that all audio clips have the same length, making them easier to process and feed into machine learning models.

• Improved computational efficiency: Some machine learning algorithms perform faster when processing data of fixed length.

• Improved model performance: Padding to the same length ensures that all audio clips have an equal opportunity to contribute to the training process, which can lead to improved model performance.

For labeling our audio files that we want to train (y_train) we faced multi attribute. Our solution is that merge our two attributes (action and object) for labeling and labelled them based on the whole action and the object that considered. [4].

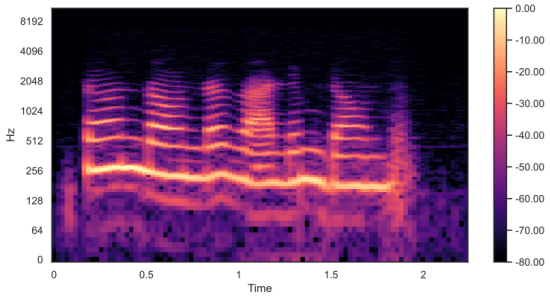

Fig. 4. Spectrogram of Sample Audio

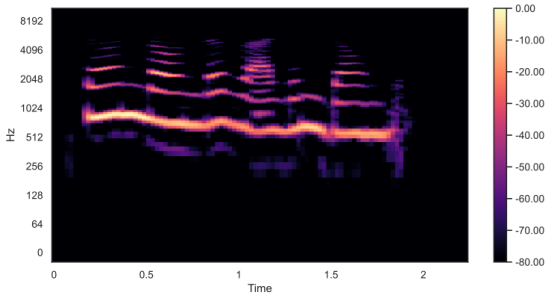

Fig. 5. Mel Spectrogram of Sample Audio

B. Model selection

Convolutional Neural Networks (CNNs) can be used for audio classification tasks, such as speech recognition, music genre classification, and sound event classification. In audio classification, CNNs can learn and extract features from the raw audio signals and classify the audio into predefined categories.

In audio classification, the input to the network is usually a two-dimensional representation of the audio, such as a spectrogram, where the first dimension represents time and the second dimension represents frequency. Convolutional layers are then used to extract features from this representation, and these features are fed into fully connected layers for classification. Pooling layers are used to reduce the spatial

resolution of the feature maps, making the network invariant to translations in time and frequency. [5].

We use Convolutional Neural Networks (CNNs) for audio classification because:

• CNNs are good at processing and analysing audio signals, which are essentially 1D signals, by considering local features and detecting patterns.

• CNNs can effectively handle the variability in the length of audio signals and still extract meaningful features.

• Pre-trained CNN models can be used as feature extractors to obtain robust representations of audio signals.

• The convolutional layers in a CNN can learn a translation-invariant representation of the audio signal.

Overall, CNNs are well-suited for audio classification tasks, making them a popular choice among researchers and practitioners [6].

C. Hyperparameters tuning

In this model we do not use the hyperparameter tuning and we reached to better accuracies with many repetitions and trials.

There are two main sets of hyperparameters can be tuned:

• top_db (threshold in decibels) for the pre-processing

• CNN parameters

Top_db refers to the threshold in decibels used to determine where to trim the audio signal. It is used as a measure of the energy or loudness of the audio signal. The audio signal is trimmed from both the beginning and the end, starting and ending where the energy of the audio signal is above the top_db threshold. The value of top_db determines the level of energy below which the audio signal will be trimmed. The higher the top_db value, the more the audio signal will be trimmed.

All over, we will train a CNN with different numbers of input layers, convolutional layers, dense layer and the dropout rate, which are some of hyperparameters belong to CNN.

III. Results

The pre-processing of raw audio like trim, padding, z-score normalization is a very important step in such a dataset that has noisy data, furthermore the recognition classifiers can be influential can be influential. For instance in this project at the begging we use a model with 60 db for trimming and a model with 1 input layer, 4 convolutional layers, 2 dense layers and 1 output layer the accuracy of the model was ≈45%. Then with 5 db and with 1 input layer, 4 convolutional layers, 4 dense layers, and 1 output layer, which are parameters of CNN we reach the accuracy ≈79%. Finally, the best result reached at 10 db for the hyperparameter of top_db was ≈ 93%.

As a result, the good dominance in the data set and the dissipation of our dataset is very important step of processing. Moreover, we need to knowledge of the performance of different classifiers based on experiences and other results. Finally, the classifier and the method of pre-processing, which we choose to project, are different on the dataset to dataset due to distribution of data and the aim of the project such as intent recognition, language detection, sentiment of speaking.

IV. Discussion

CNNs are designed to process grid-like data, such as images. In the context of audio classification, CNNs use convolutional layers to extract features from spectrograms or other audio representations. The convolutional layers are designed to recognize patterns in local regions of the input data, and the features extracted by the CNN can be used to classify the audio.

On the other hand, RNNs are designed to process sequential data, and are well-suited for tasks such as speech recognition where the context of previous frames is important in determining the correct output. RNNs process each sample in a sequence one at a time, and use the hidden state from the previous sample to inform the processing of the current sample [7].

One reason why CNNs are often considered better than RNNs for audio classification is that they are faster and more computationally efficient. CNNs can also be trained on much larger datasets, as they do not require sequential processing of the data. Additionally, CNNs have shown excellent performance on a wide range of audio classification tasks, including speech recognition, music classification, and sound event detection. However, the choice between RNNs and CNNs for audio classification ultimately depends on the specific requirements of the task. In some cases, an RNN may still be the best choice due to its ability to model sequential dependencies in the data.

Overall, CNNs have been shown to be effective for audio classification, achieving high accuracy on a variety of audio classification tasks, including speech recognition, music genre classification, and sound event classification.

References

[1] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 4580–4584, Ieee, 2015.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[3] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *2015 IEEE 25th international workshop on machine learning for signal processing (MLSP)*, pp. 1–6, IEEE, 2015.

[4] N. Takahashi, M. Gygli, B. Pfister, and L. Van Gool, "Deep convolutional neural networks and data augmentation for acoustic event detection," *arXiv preprint arXiv:1604.07160*, 2016.

[5] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, *et al.*, "Cnn architectures for large-scale audio classification," in *2017 ieee international conference on acoustics, speech and signal processing (icassp)*, pp. 131–135, IEEE, 2017.

[6] Q. Mao, M. Dong, Z. Huang, and Y. Zhan, "Learning salient features for speech emotion recognition using convolutional neural networks," *IEEE transactions on multimedia*, vol. 16, no. 8, pp. 2203–2213, 2014.

[7] G. Parascandolo, H. Huttunen, and T. Virtanen, "Recurrent neural networks for polyphonic sound event detection in real life recordings," in *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 6440–6444, IEEE, 2016.