Speech Recognition with Convolutional Neural Network

Giulia Chillemi, Valeria Bo
Politecnico di Torino
Student id: s296737, s304645
s296737@studenti.polito.it, s304645@studenti.polito.it

***Abstract*—This project aims to build an accurate Speech Command Recognition system, that is capable of detecting predefined seven vocal commands. Using the Speech Dataset located at this link[1] we have implemented a convolutional neural network machine learning algorithm with MFCC features as input.**
The proposed approach obtains overall satisfactory results giving us a good performance and achieving an accuracy of 89.09% .

I. Problem overview

The aim of the problem is to predict the intent of an input audio sample, where the intent is defined by an action and an object.
The dataset consists in a collection of audio file in a WAV format divided in two parts:

- A development set containing 9854 recordings
- An evaluation set containing 1455 recordings

Each record of the development set is characterized by:

- path: the path of the audio file
- speakerId: the id of the speaker.
- action: the type of action required through the intent.
- object: the device involved by intent.
- Self-reported fluency level: the speaking fluency of the speaker.
- First Language spoken: the first language spoken by the speaker.
- Current language used for work/school: the main language spoken by the speaker during daily activities.
- gender: the gender of the speaker.
- ageRange: the age range of the speaker.

We will use the development set to build a classification model to correctly label the records in the evaluation set. The development set is not balanced: in particular the labels with the smallest and the biggest number of recordings are ”deactivate lights” and ”increase volume”, respectively with 552 and 2614. All the records are mono channel and with a sample width of 16 bits.
Some of the sounds files are sampled at a sampling rate of 22050Hz, while most of them have a sampling rate of 16000Hz. This means that 1 second of audio will have an array size of 22050 for some sound files, while it will have a smaller array size of 16000 for the others. Recordings duration differs from record to record and we can see a representation of the distribution of the duration in Fig1.

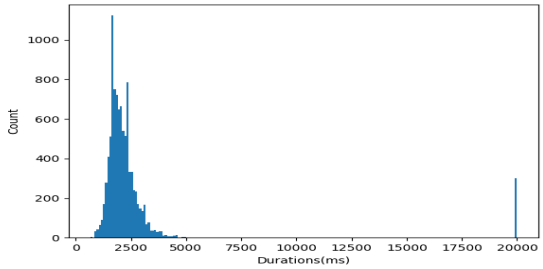

Fig. 1. Distribution of the duration of recordings

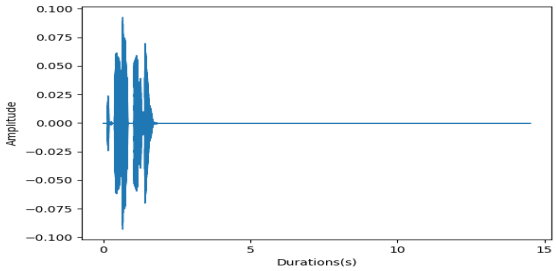

Fig. 2. Representation of a 20s long recording

We can see that there are some outliers with a length of 20000 ms but after a manual inspection we found out that these records contain some silence after the pronounced command as we can see in Fig2.

II. Proposed approach

A. Preprocessing

As illustrated in the previous section the sample rate is not uniform for all the sounds file in the dataset. For this reason we have standardized and converted the audio to the same sampling rate of 16000Hz to have all the arrays with same dimension.
We have also resized the audio samples to have the same length (3 second) by either extending its duration by padding it with silence, or by truncating it.
In this study, the features are extracted using the Mel Frequency Cepstrum Coefficient (MFCC) that is used to indicate the spectral envelope that helps to understand the tone and

[1]https://drive.google.com/file//1gUPpqPTlgfzIyDU4eG6t5HoTlK5inLO4/view?usp=sharing

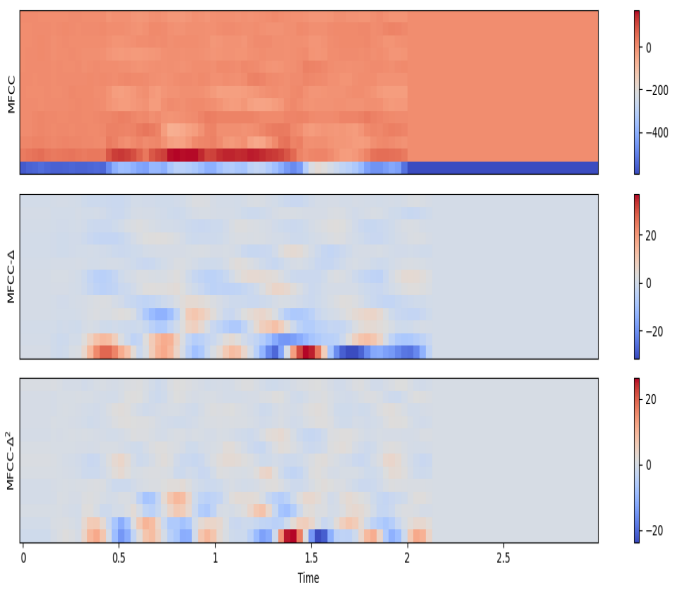

Fig. 3. Mel-frequency cepstral coefficients (MFCCs)

sound structure of a human voice [1]. MFCCs are created by applying cepstral analysis to the log mel-spectrogram. The cepstral analysis is an analyzing technique that use log-scaled spectrum with inverse Fourier transforms. After converting raw audio data into spectrogram via Short Time Fourier Transform, a Mel-Frequency filter bank is applied to make spectrogram perceived similar to the human auditory. Then, is used a log scale to perform cepstral analysis and obtain MFCCs by performing Discrete Cosine Transform.
Using Python library librosa[2] the MFCC features are extracted using a window of size 1024 with hop-size of 512 and stored in an array. Then, using again librosa, we compute the first and second derivatives represented in Fig3.
The number of coefficients follows the guidelines found in [2]: to recognize voice command, the first 13 MFCC coefficients are employed, together with their first and second delta values, forming a total of 39 MFCC coefficients.
With this parameters we obtain a matrix of 39 × 94 for each audio.

B. Model selection

Convolutional Neural Networks (CNNs) have been widely used in the field of speech recognition and classification, since they often provide positive results. More recently, experimental results show that CNNs reduce the error rate by 6%-10% compared with DNNs on the TIMIT phone recognition [3]. We report here some key layers and their function:

- Convolutional layer (Conv2D) unit is the portion that learns the translation invariant spatial patterns and their spatial hierarchies
- Max Pooling layer down-sample by taking max operation to reduce the amount of parameters
- Dropout layer control overfitting by randomly setting the weights of a portion of the data to zero
- Dense layer contains hidden layers tied to the degrees of freedom that the model has to try and fit on the data
- Flatten layer squishes all the feature map information into a single column in order to feed it into a Dense layer, where the outputs of the last one are the 7 labels that the model is supposed to classify the audio recordings into.

Shallower and deeper variations of a CNN have been implemented, informed by speech recognition model in [2].

The proposed CNN architecture is represented in Fig4. This is fed with the Mel-Frequency Cepstral Coefficients (MFCCs) extracted from audio signals and its time derivative.

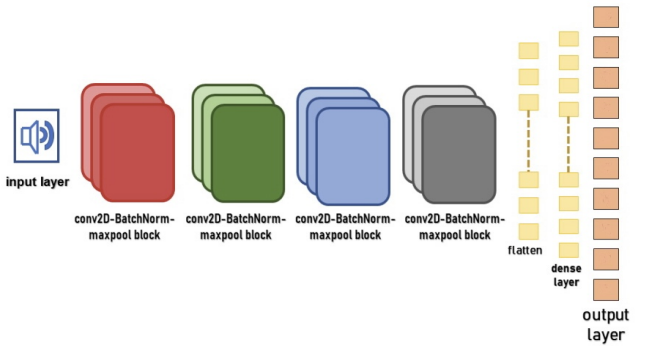

Fig. 4. CNN architecture

The input size of the model is (14637 × 39 × 94 × 1), where 14637 is the number of input sample data after the oversampling, 39 is the number of feature dimensions, 94 is the number of frames per sample, and 1 is the depth of the channel per sample. The preprocessed train data are put into a 2D convolutional layer with a filter size of 24 and a kernel size of (3, 3). This input layer also uses the reLu activation function and an L2 regularizer with a regularization factor of 0.1.
After a BatchNormalization layer's the output is transferred to the MaxPooling layer which has a pool size of (2, 2). This sequence of layers is repeated other three times, with filters size of 32, 64, and 128 in each Conv2D layer. The output vector is converted to a one-dimensional vector by the flatten layer and the 1D vector is then connected to a 128-unit fully connected dense layer using a reLu activation function.
A Dropout layer is used to avoid overfitting while the softmax function is used in the classification process.
Categorical crossentropy and Adam [4] with a learning rate of 0.0001 are employed as, respectively, loss function and optimizer. The best-working configuration of the CNN is identified through the search described in the following section. The learning rate is set to 0.0001 while the number of batch size is equal to 32. The model is trained for 40 epochs.

[2]Available at: https://librosa.github.io/

C. Hyperparameters tuning

For evaluating our model and asses hyperparameters values we have split our development set in a training and a test set with a proportion of 80-20 respectively.
Since our dataset is imbalanced, a technique of random over-sampling is applied to our training data over all the minority classes in order to have the same size of records as the majority class.
After the first performance, the results gave us a problem related to overfitting with a high accuracy on the training set but a poor accuracy on the validation set.
In order to cope with this, we have trained our model on different values of dropout increasing the rate from 0.1 to 0.4. The results are improved, as we can seen in Fig5 and Fig6. Nevertheless, the overfitting problem is still present so we have decided to add dropout layer's at the end of each pooling layer starting with a rate of 0.1 and then 0.2.

III. Results

After the previous research, the best configuration for the CNN is illustrated in Table 1.

Layer Type	Characteristics	Output Shape
Convolution + ReLu + l2regularization	n°filters=24, kernel size= 3x3	(None,37,92,24)
BatchNormalization		(None,37,92,24)
MaxPooling2D	pool size=2x2	(None,18,46,24)
Dropout	rate=0.2	(None,18,46,24)
Convolution + ReLu + l2regularization	n°filters=32, kernel size= 3x3	(None,16,44,32)
BatchNormalization		(None,16,44,32)
MaxPooling2D	pool size=2x2	(None,8,22,32)
Dropout	rate=0.2	(None,8,22,32)
Convolution + ReLu + l2regularization	n°filters=64, kernel size= 3x3	(None,6,20,64)
BatchNormalization		(None,6,20,64)
MaxPooling2D	pool size=2x2	(None,3,10,64)
Dropout	rate=0.2	(None,3,10,64)
Convolution + ReLu + l2regularization	n°filters=128, kernel size= 3x3	(None,1,8,128)
BatchNormalization		(None,1,8,128)
MaxPooling2D	pool size=2x2	(None,1,4,128)
Dropout	rate=0.2	(None,1,4,128)
Flatten		(None,512)
Dense + ReLu	n°hidden units=128	(None,128)
Dropout	rate=0.4	(None,128)
Dense + Softmax	n°hidden units=7	(None,7)

TABLE I

In Fig7 is represented the final model accuracy.
Increasing the number of epochs to 70 the CNN reaches a validation accuracy score of 0.8909.
Finally we can visualize the confusion matrix of the model in Fig8. As the confusion matrix shows, the problem with our model remains the predictions of classes that differ just for first syllable of the vocal command such as "increasing/decreasing volume" and "increasing/decreasing heat".
All the previous results have been obtained with tensorflow (version 1.15.0) and keras (version 2.2.4).

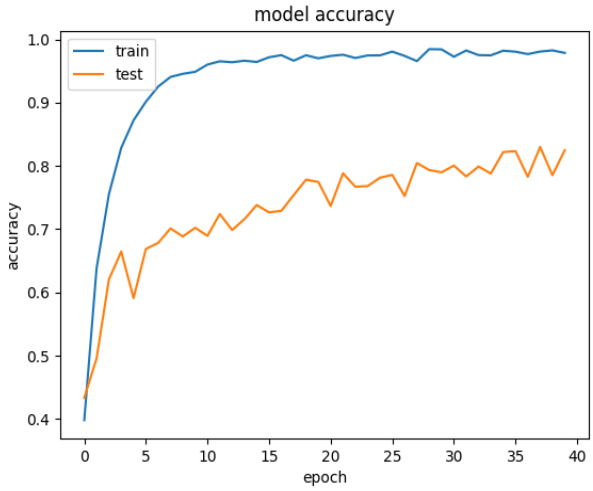

Fig. 5. Dropout=0.1

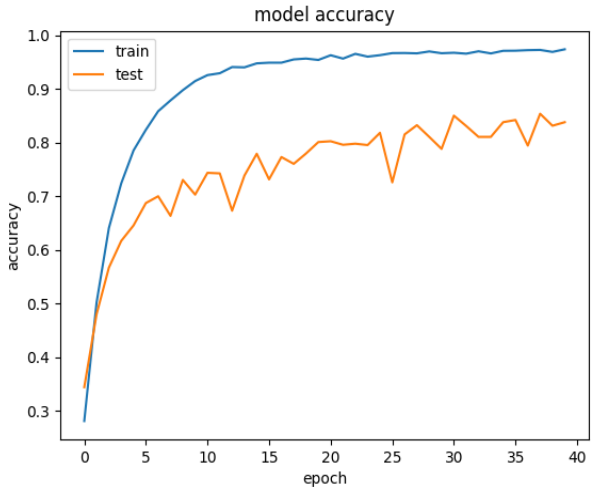

Fig. 6. Dropout=0.4

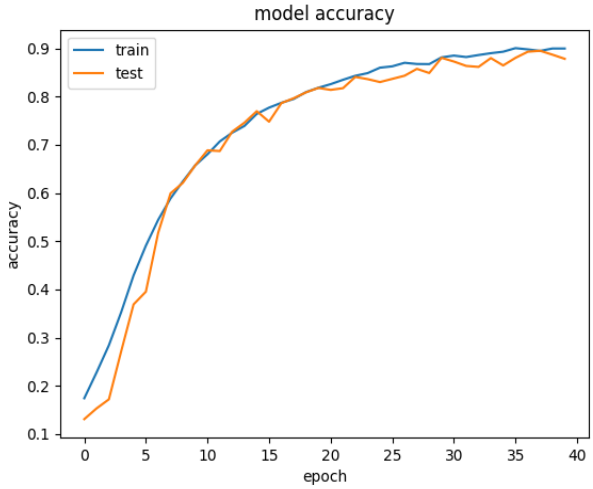

Fig. 7. Final model accuracy

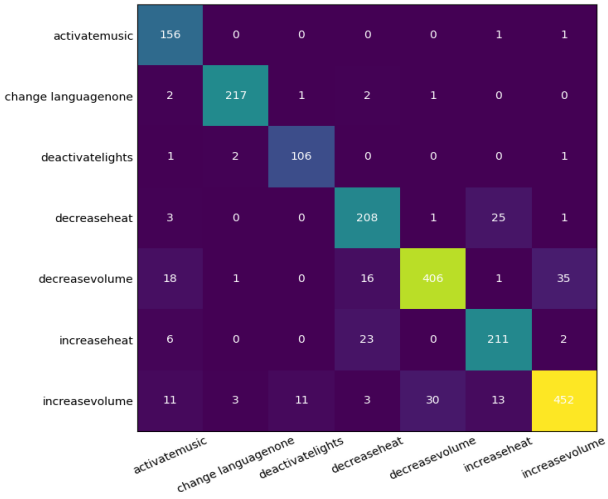

Fig. 8. Confusion matrix

IV. Discussion

Since CNN requires a large amount of data for the purpose of learning, the more data is given, the better and more accurate the classification accuracy of the CNN will be.
Other changes that could lead to a better generalized performance of our CNN are training the model on a dataset with more different speaker's accents in order to range the input sound pronunciations, improving our feature extraction adding some noise in the data records [5] [6] [7] and also combining different types of features [8].
Some studies also show that the accuracy can improve by adding a SK regularization, reaching a value 55% higher [9]. Finally, a better configuration of hyperparameters values can be achieved with a Grid Search validation method but, for the high computational cost, it has not been tackled in this report.

References

[1] J. B. Youngsik Eom, "Speech emotion recognition using 2d-cnn with mel-frequency cepstrum coefficients," *Journal of information and communication convergence engineering (JICCE)*, vol. 19, no. 3, pp. 148–154, 2021.
[2] P. R. Ovishake Sen, Al-Mahmud, "A convolutional neural network based approach to recognize bangla spoken digits from speech signal," *International Conference on Electronics, Communications and Information Technology (ICECIT)*, 2021.
[3] H. J. L. D. G. P. Ossama Abdel-Hamid, Abdel-rahman Mohamed and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING*, vol. 22, no. 10, pp. 1533–1545, 2014.
[4] S. Doshi, "Various optimization algorithms for training neural network," *Towards Data Science*, 2019.
[5] C. M. Bishop, "Training with noise is equivalent to tikhonov regularization," 1995.
[6] R. J. M. Russell Reed, "Neural smithing: Supervised learning in feedforward artificial neural networks," 1999.
[7] C. M. Bishop, "Neural networks for pattern recognition," *Oxford University Press, USA*, 1996.
[8] D. V. T. Urmila Shrawankar, "Techniques for feature extraction in speech recognition system : A comparative study,"
[9] B. M. L. Reuben Feinman, "Learning a smooth kernel regularizer for convolutional neural networks,"