Deep Learning for Audio Classification: An Overview and Analysis

Gaetano Salvatore Falco
Politecnico di Torino
Student id: s280209
gaetanosalvatore.falco@studenti.polito.it

Kuerxi Gulisidan
Politecnico di Torino
Student id: s304915
kuerxi.gulisidan@studenti.polito.it

***Abstract*—In this report, we propose a deep learning-based method for classifying audio recordings based on the intent expressed in them. The proposed approach utilizes a Depthwise Separable Convolutional Neural Network (DS-CNN) to predict both the requested action and the object affected by the action. Our method begins with standardizing and preprocessing the input data, which is crucial for achieving high accuracy in the model. We then propose a CNN model to classify intents in the evaluation set. The performance of our model is evaluated using accuracy, and the results demonstrate the effectiveness of the proposed approach.**
All code is publicly available at : https://github.com/chowned/Audio-recognition-for-IOT

I. Problem overview

During the last decade, there has been a growing interest for developing methods to automatically classify audio recordings based on the intent expressed in them, with real applications ranging from audio event detection for baby monitoring systems or in the game industry. For this project, we will work on an intent detection problem given an input audio sample. We aim to predict both the requested action and the object that is affected by the action. The dataset is divided into the following parts:

- **audio**: containing 11309 WAV format audio files recorded by 97 speakers.
- **development.csv**: a comma-separated values file containing the records from the development set.
- **evaluation.csv**: a comma-separated values file containing the records corresponding to the evaluation set.

This problem is made even more difficult by the variability in the way that different speakers express the same intent.

When reviewing the development.csv file, we can see that it contains 10 columns and 9854 rows, leveraged to obtain the labels required for training and validating models. The evaluation.csv file contains 8 columns and has 1454 rows, lacking the action and object columns that have to be predicted.

The provided dataset is characterized by the following columns:

Id
path
speakerID
action
object
self-reported fluency level
first language spoken
gender
ageRange
current language used for work/school

Our first task was to detect the audio files to use from the column ”path” and assign an Identifier and a Label. The latter will be composed as the concatenation of the ”Action” and the ”Object” columns.

II. Proposed approach

A. Automatic Audio length finder with given threshold

After some data exploration, we found out that the provided dataset is composed of audio files with different lengths, according to Fig.1.

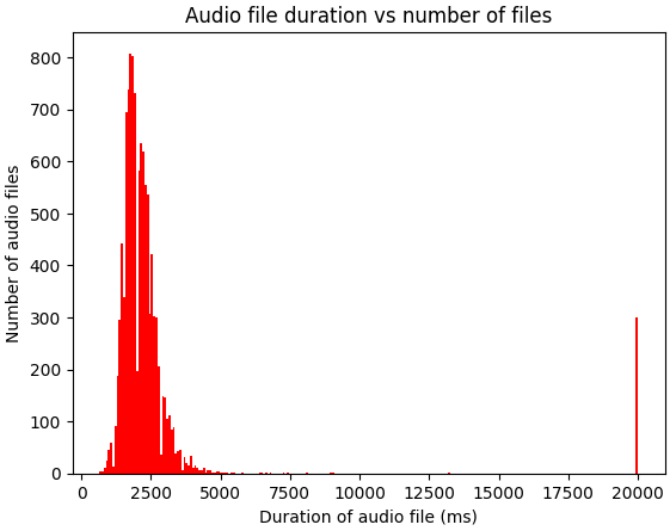

Fig. 1: Audio file duration

As we have to build a model and specify the input shape, we found it necessary to standardize our input data. We developed the ”find_duration” function to accomplish this task and, provided a threshold of 90%, it returns the audio length that is able to incorporate such a percentage. This length was saved as a variable in seconds and used to standardize the length of the entire dataset through padding.
While the audio length in the dataset was spread as a normal distribution with outliers, we chose to use the ”find_duration” function instead of calculating the mean and standard deviation as it is not dependent on the initial distribution.

B. Data Preprocessing

In this section, we describe the preprocessing steps taken on the audio files to normalize and standardize them for the model.
We make sure that all audio files share a common ground in terms of sampling rate and duration by using the preprocess function.

- **Sampling rate**: 8.000, this gives us a good balance between audio quality and prediction time.
- **Duration**: 4s, this is automatically found by the ”find_duration” function.

We use the librosa library to trim the audio files, remove silence and change the sampling rate if it is different by calling the ”process_file” function according to these steps:

- **Label building**: as we have to predict both the object and the action, we use this information to build the corresponding label for each audio file.
- **Audio trim**: We remove silence from the audio files. This could be a huge problem if we start by cutting the length at 4s and our speaker starts to talk after 3 seconds.
- **Resample and padding**: as there are different sampling rates in the provided dataset, we make sure the output sampling rate is 8.000 and, if the input audio file after the trim operation is less than 4s, we increase its duration by using padding and make it compliant with our choices.
- **Storing the audio files**: as we want to fine-tune the hyperparameters of our model, applying these steps at each run is a huge workload. We then decided to save both the development dataset and the evaluation dataset into two folders, which will be called by the model for the prediction.

We eventually proceed to call the ”preprocess” function on the development and evaluation set, which will perform these actions:

- **Input audio to mono channel**: we decide to convert all of our audio files to a single channel, as this is more than enough to store the necessary information.
- **STFT**: we apply a short-time Fourier transform to convert the audio files from the time domain to the frequency domain, and then we compute the spectrogram by taking the absolute value of the STFT result.
- **MFCCs**: we have chosen an audio representation based on extracting the features as Mel-frequency cepstral coefficients. This allows us to carry the most valuable information while also being able to use convolutional layers in our model.

In the following figures, an example of the audio stream as is passed to the model.

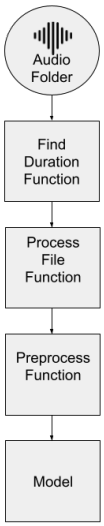

Fig. 2: Stream for the data to enter in the Model

In Fig.3 there is an example of the audio with the label ”activate music” while in Fig.4 an example of the audio with the label ”increase heat”.

These examples also allow us to show the results of the padding function, and to better show that no information is carried to the model that can focus only on the part that carries the information.

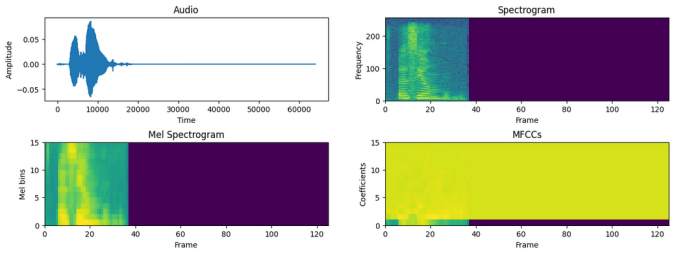

Fig. 3: ”1-activatemusic”

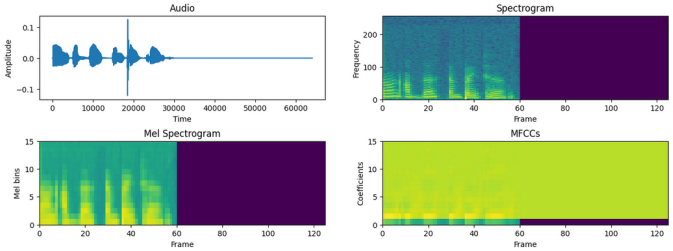

Fig. 4: ”6-increaseheat”

C. Model selection

Our approach to audio classification is mainly based on the researches done by Palanisamy et al. [1] and Peter Mølgaard Sørensen et al. [2].
While the final goal and approach of [1] was different than ours, as it proposed a method of using transfer learning on the weights instead of random initialization for audio classification, we chose to follow their approach to make the model learn the energy distributions in the spectrograms.
On the other hand, [2] proposed a depthwise separable convolutional neural network for keyword spotting. Their approach has been proven to achieve state-of-the-art performance by utilizing the Mel-frequency cepstral coefficients (MFCCs) technique, which is implemented in the provided code for data preprocessing. MFCCs allow us to extract relevant features from the audio data and uses them to train a CNN model that is fine-tuned for the task of audio classification. Overall, this approach is a promising way to achieve high accuracy in audio classification tasks.
Moreover, the presented algorithm has proven to achieve high accuracy also after an heavy pruning was applied, as shown in the following sections.

D. Hyperparameters tuning

The training of the Depthwise Separable Convolutional Neural Network (DS-CNN) was carried out using a learning rate of 0.01, a batch size of 32, and 512 output filters in the convolution. The number of filters was reduced to speed up the training process by multiplying it with an alpha of 0.125. The following hyperparameters were optimized to obtain the best combination:

- **Frame length**: set to 0.064 seconds
- **Frame step**: set to 0.064 seconds
- **Lower frequency**: set to 20 Hz
- **Higher frequency**: set to sampling rate/2 = 4,000 Hz

This research aimed to explore the feasibility of using a small CNN for audio classification, and despite the limited resources, the team was able to optimize the parameters for building the MFCCs to achieve high accuracy while keeping a low computational time. The results suggest that this approach could be suitable for IoT devices that have limited computing power but further research is needed to fully understand the potential of this approach.

To monitor the training and evaluate the results, TensorBoard was used along with different callbacks. The rest of the network is initialized with Xavier initialization, and all parameters are updated during training with the Adam optimizer while a linear decay was set for the learning rate. The training was early stopped when the accuracy on the training set reached 100%.

III. Results

We tested our model with different batch sizes and alpha values and the results are shown in TABLE I. Increasing alpha value has shown to give better results, at the expense of being more time-consuming.

Hyper-parameters Grid Search		
Alpha	Batch size	Test Accuracy
0.125	64	0.885
	32	0.889
	16	0.920
	8	0.907
	4	0.889
1	64	0.940
	32	**0.959**
	16	0.951
	8	0.945
	4	0.956

TABLE I

The best configuration for the model without pruning was found with a Batch Size of 32, but the resulting number of 4.739.079 trainable parameters means the train takes a huge amount of time. This has been reduced by applying pruning with the alpha parameter, thus by reducing the trainable parameters to 76.295. This approach was still able to achieve an high performance as shown in the table for a Batch Size of 16.

IV. Discussion

In this report we propose to explore the potential of using a small CNN for audio classification and to highlight the impact of proper hyperparameter tuning on its performance. The results demonstrate the suitability of a CNN for audio classification even with limited computational resources.

As we had limited resources and time, we were not able to test our approach further and provide more results. Nevertheless, we believe these experiments are a valuable starting point for future research in this field, as our goal was never to achieve the highest performance but to provide a good balance between accuracy and computational costs.

In conclusion, our work highlights the potential of using small CNNs for audio classification and we hope that this sparks future research in this field, as it could lead to even better results.

References

[1] Kamalesh Palanisamy, Dipika Singhania, Angela Yao, "Rethinking CNN Models for Audio Classification" in arXiv 2020 (https://arxiv.org/pdf/2007.11154.pdf)

[2] Sørensen, P.M., Epp, B. & May, T. "A depthwise separable convolutional neural network for keyword spotting on an embedded system." J AUDIO SPEECH MUSIC PROC. 2020, 10 (2020). (https://doi.org/10.1186/s13636-020-00176-2)

[3] Mingwen Dong, "Convolutional Neural Network Achieves Human-level Accuracy in Music Genre Classification" in arXiv 2018 (https://arxiv.org/pdf/1802.09697.pdf)

[4] Pete Warden, "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition" in arXiv 2018 (https://arxiv.org/pdf/1804.03209v1.pdf)