Comparative Analysis of Deep Learning Models for Environmental Sound Classification ESC-10

De Luca Chiara[†], Gianfrate Flavia[‡]

Abstract—**This project aims to explore and compare different models and techniques applied to the ESC-10 dataset, composed by 10 distinct sound classes. Specifically, we will investigate the effectiveness of using spectrograms and MFCC as audio representations and evaluate the performance of models such as CNN, RNN, and CRNN. Additionally, we will employ pruning as a powerful model optimization approach to enhance efficiency without compromising accuracy. Through a comprehensive analysis and comparison with related studies, we will provide valuable insights into the application of these methods in sound classification tasks.**

I. Introduction

In this paper, we address the problem of classifying 10 environmental sound classes using deep learning techniques. Our objective is to compare the performance of different deep learning architectures on this classification task. Specifically, we investigate the effectiveness of Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Combined CNN-RNN architectures (CRNN) using Mel-Frequency Cepstral Coefficients (MFCC) data. Additionally, we explore the performance of a CNN model trained on spectrogram images. To evaluate the effectiveness of the different deep learning models, we compare their performance on the 10 environmental sound classes. We utilize the ESC-10 dataset containing diverse sound recordings from various environments. In addition to comparing the performance of our proposed models, we also benchmark our results against a well-established paper in the field written by Zohaib Mushtaq and Shun-Feng. This paper is organized as follows: sections 2, 3 and 4 consist of data exploration and data preprocessing, i.e. they provide a brief description of the dataset, how we can represent sound data and useful methodologies to make the data ready for the modeling part. In section 4 in particular we will describe theoretically with the help of our data, the two approaches used in the paper, namely the use of MFCC and spectrograms. Sections 5 and 6 are those dealing with modeling. In the first from a theoretical point of view, in the second we will describe the configurations of the models that we will use for the classification problem and their results. At the end we then have a section dedicated to the comparison with the paper mentioned above, a section on the comparison of the methodologies and results obtained and a concluding section.

[†]Department of Mathematics, University of Padova, email: {chiara.deluca.1}@studenti.unipd.it

[‡]Department of Mathematics, University of Padova, email: {flavia.gianfrate}@studenti.unipd.it

Special thanks / acknowledgement go here.

II. Dataset

ESC-10 [1] is a subgroup of the well-known ESC-50 dataset, widely used in the field of sound classification. The ESC-10 dataset is composed by 10 different classes of environment sounds:

0	1	2	3	4
chainsaw	clock˙tick	crackling˙fire	crying˙baby	dog
5	6	7	8	9
helicopter	rain	rooster	sea˙waves	sneezing

Each audio lasts 5 seconds. The data are balanced, there are 40 sounds for each type, for a total of 400. Typology of the sounds is quite different from each other, and also the temperament of the sounds can be grouped into different classes [1]:

- More or less structured noise/soundscapes (rain, sea waves, fire crackling, helicopter, chainsaw);
- Percussive sounds (sneezing, dog barking, clock ticking);
- Harmonic sounds (crying baby, crowing rooster).

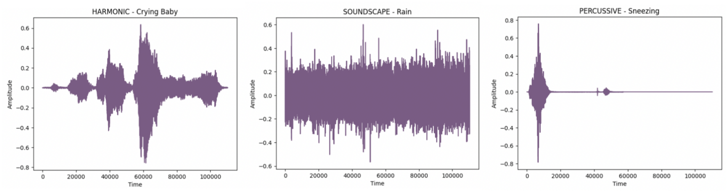


Fig. 1: Audio signal for each typology

Sounds in the dataset are very accurate and clearly recognizable by the human ear. Given the small number of data, the use of data augmentation techniques was necessary in order to get a more robust and precise type of classification model. The structure of ESC-10 data configures the classification problem of a slightly different nature than the one applicable on the ESC-50 dataset, as the nature of the sounds is more varied and less confusing. This is the reason why the accuracy that will be demanded from a good model is higher.

III. Data Pre-Processing and EDA

As mentioned before, the first thing to do is to apply some transformations to each sound in order to have augmented data. More specifically, the techniques implemented are:

1) ***Noise Addiction***: adding white noise to the sound
2) ***Time Shifting:*** moving the sound over time
3) ***Time Stretching:*** speeding up the sound

4) ***Pitch Shifting:*** raising or lowering the pitch of a sound

Moreover, for a greater increase in data, we have implemented functions aimed at combining these approaches with each other. These functions need to know some parameters that characterized these transformations. We opted for a 0.05 noise addiction, a time shifting of 50000, a time stretching of 2 (halves the duration of the audio), and a pitch shifting of 5. Then we combine some of these simple transformations in this way:

5) Noise Addiction + Time Shifting
6) Time Stretching + Pitch Shifting
7) Noise Addiction + Pitch Shifting
8) Time Stretching + Time Shifting

Adding the single use and the combined use of the approaches, we obtain other 8 different versions of the same starting audio.

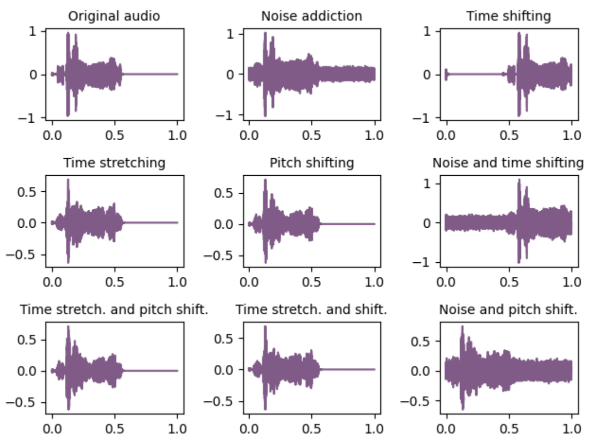


Fig. 2: Waveform of augmented data

This allows us to continue with the implementation of Deep Learning models that will presumably be more robust. The dataset that will be used for the development of the model will therefore consist of 3600 audios. An important precaution is to resize to 5 seconds the data to which time stretching has been applied, we can do this through adding zeros. If we don't do this, we'll have problems building the models because we need data of the same size.

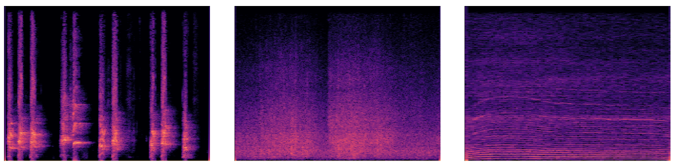
Once we have our final dataset, we proceed with the normalization of the data and the creation of the variable y which represents the 10 audio categories. As regards the creation of X, we dedicate the next section to the description of 2 different approaches used for the classification and therefore for the creation of our features.

IV. Audio Classification Approaches

In order to well classify sounds, we try to implement two main different approaches:

A. Spectrogram: the visual sound

A spectrogram, by definition, is a visual representation of the spectrum of frequencies of a signal as it varies with time. It is computed thanks to the technique of Fast Fourier Transform, applied to each short segment of the initial audio. The result of these steps is a plot that puts in correlation time and frequencies. The typical colors of the spectrogram are used to identify energy and amplitude of the sound at each point of the plot. In order to classify audio with this approach, you will need to use image classification techniques (ex. CNN).



(a) Percussive sound (b) Soundscape (c) Harmonic sound

Fig. 3: Spectrograms for each type of sound

B. Mel Frequency Cepstral Coefficients

MFCCs are small set of features which try to describe the shape of the spectral envelope of a sound. They are widely used in the field of speech recognition and can also be very useful for audio classifications. They are based on the mel scale, a scale obtained through a non-linear transformation of sound frequencies. The principle on which the transformation is based is to make the sound closer to the type of sound perceived by the human ear. That's why the transformation tends to emphasize the low pitches and flatten the high pitches. Below the Mel Scale equation:

$$M(f) = 2595 \cdot \ln\left(1 + \frac{f}{700}\right) \tag{1}$$

Let's take a look at how they are obtained:

- Frame the signal
- Compute the power spectrum for each frame through FFT
- Apply Mel-filterbank to power spectrum
- Take the logarithm of the filterbank energies to convert them into decibels (log-scale)
- Discrete Cosine Transformation computation on the log-filterbank to get MFCC coefficients

Below is the image of the audio signal transformed into MFCC, passing through the Mel Spectrogram.

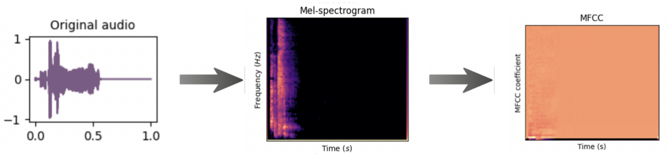


Fig. 4: Roster sound: from audio to MFCC

We implemented the librosa MFCC code using log-power Mel spectrogram of each sound with their correspondent sample rate. Furthermore, these values for main MFCC parameters have been chosen [2] :

- number of MFCC: 60
- hop lenght: 512
- win length: 1024

When using MFCC, the chromagram can be used as an additional input. The chromagram is simply the graphical representation of the frequency spectrum of an audio signal over time, and its use is particularly suitable when there are musical and tonal features that can aid in classification. Therefore, we have chosen to incorporate it, hoping that our model can benefit from it.

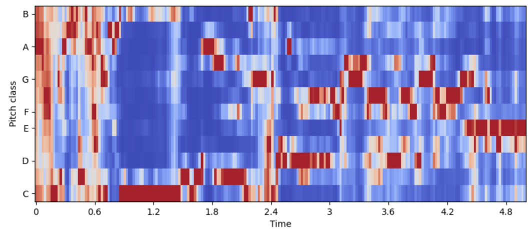


Fig. 5: Chromatogram of a Crying Baby sound

C. *Spectrogram VS MFCC: what is better?*

It is evident that both approaches are very suitable for classification, even if they start from different principles. However, we can recognize notable differences from the point of view of information obtainable from each of the two approaches:

■ **Spectrogram**: more sensitive to:
- spectral component
- main sound traits (pitch, timbre...)

■ **MFCC**: less sensitive to:
- noise
- temporal information

It is clear that the two approaches have pros and cons within our classification. Indeed, given the nature of our sounds (of different timbres and musical connotations), a spectrogram might be more useful. However, MFCCs may prove much more capable of classifying augmented data, which has been generated with added noise and time changes. In conclusion, to determine which is the best approach, it is necessary to experiment.

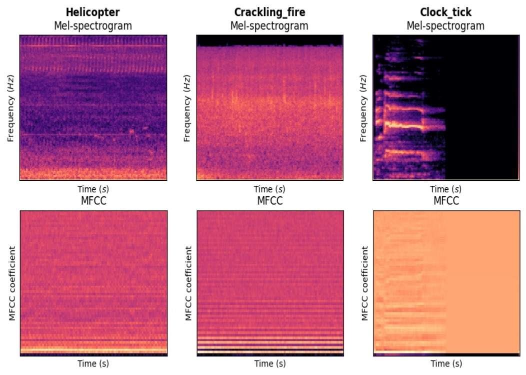


Fig. 6: Spectrograms and Mfccs compared for sounds of three different natures

V. A Short Networks Methodology Overview

A. *Recurrent Neural Network*

An RNN is a neural network with the aim of processing sequential data. The presence of loops allows us to keep sequences of information that could be very useful to correctly classify data. More specifically, we implemented a Long Short-Term Memory, a type of (RNN) architecture designed to address the vanishing gradient problem. LSTMs in music and sound classification have proven to be effective in a variety of applications (especially in speech recognition). Indeed, LSTMs can be used to process sequential data, such as spectrograms or Mel Frequency Cepstral Coefficients (MFCCs). We will use this methodology for MFCC approach.

B. *Convolutional Neural Network*

Convolutional Neural Networks (CNNs) have revolutionized many areas of machine learning, including computer vision and speech and audio processing. One specific application of CNNs in audio signal processing is the classification of sounds using spectrogram images. CNNs are particularly suited for this task because they are designed to learn hierarchical representations of spatial features. In the case of spectrograms, the frequency axis corresponds to the spatial dimension, and the time axis corresponds to the temporal dimension. This makes CNNs well-suited to extract relevant features from spectrograms and learn to classify audio signals based on those features. Additionally, CNNs have also been shown to be effective in the classification of other types of data that are not images. CNNs can also use MFCCs data that can be viewed as a representation of the spectral content of the audio signal. This representation is ideal for CNNs, as they can effectively learn and extract relevant features from them. Convolutional Neural Network are really useful for audio classification tasks because they are able to learn complex and non-linear relationships between the input data and the output labels and the relationship between the features of an audio signal and the corresponding label can be highly non-linear.

C. *Convolutional Recurrent Neural Network*

Convolutional Recurrent Neural Networks (CRNNs) have gained popularity in recent years due to their ability to effectively classify sequences of data, such as audio or speech signals. The reason why they are particularly suited for audio classification is that they combine the strengths of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). The Convolutional component of the CRNN is responsible for extracting relevant features from the input data, such as the MFCCs. The Recurrent component then processes the sequence of extracted features and learns to capture the temporal dependencies between them. This allows the CRNN to effectively classify audio signals based on their temporal features, which is critical for accurate classification.

VI. Experiments

In this paragraph we will present some of the best models obtained, according to the accuracy scores and the performance of the model in general.
All models have in common that the activation function is ReLU for all layers except the last one where the output is defined through a softmax. Then, 100 epochs were involved for each model, with the help of Early Stopping to prevent excessive overfitting for the model. The chosen batch size depends on the model, but in general values 32 and 64 have been found to be the most suitable. Moreover, before proceeding with the training, "categorical crossentropy" was specified as loss function, "adam" as optimizer and accuracy as metric.
The following subsections present tables that explain the configuration of the models used, specifying the type of layer added and its output size. We then continue with the training that we will observe through loss functions and accuracy functions, and finally we can do an evaluation of the various models on the test set in general and on the various classes in particular.

A. *CNN of MFCCs*

One of the best convolutional neural networks resulting from an empirical research through hyperparameters is the one formed by 3 convolutional layers with dimensions and filters shown in the following table. In particular, a regularization penalty L1 of value 0.01 was added for the last convolution layer, while padding was not used for both the convolution and pooling layers.

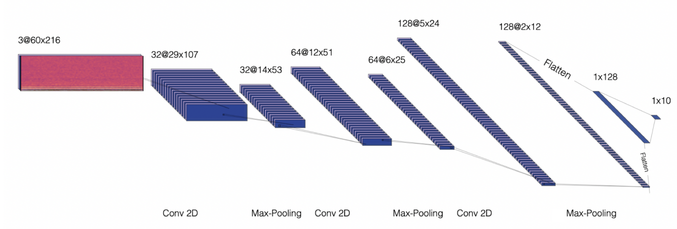


Fig. 7: CNN with MFCC: model architecture

This model has 446,922 parameters. Once the model has been set up and the loss function, metric and optimizer specified, training can begin. In the images below we can observe the trend of loss functions and accuracy functions for the train and the validation sets.

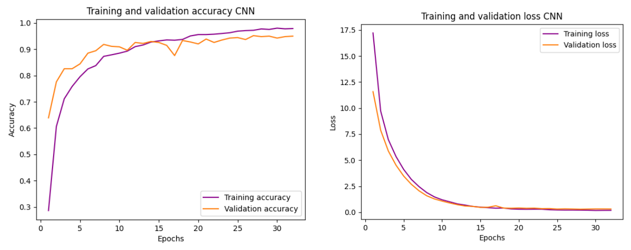


Fig. 8: CNN with MFCC: train and val accuracies and losses

The accuracy function for the training set shows a constant increase throughout the epochs without sudden falls. This indicates a gradual and continuous improvement in the model's performance over time. In the case of the validation set, we observe fluctuations in accuracy, with some peaks and valleys that can be attributed to random variations in the data or changes in model complexity. Despite this, both the two functions grow until reaching, around the 30th epoch, accuracy values not too distant from each other (about 96 % for train and 95 % for validation). As far as losses are concerned, it can be noted that the functions are very close to each other. A perfect gradual and continuous decrease is evident for both of them. Furthermore the absence of fluctuations suggests a stable and effective training process.
The model does not disappoint even on data that it has never seen, those of the test set. In this case it correctly classify 95 % of the sounds.

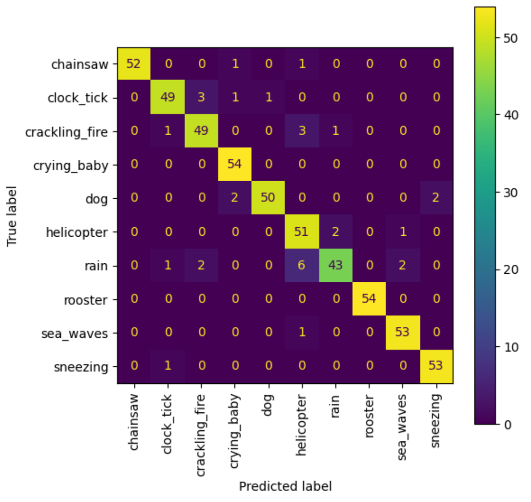


Fig. 9: CNN with MFCC: confusion matrix

In the figure above we have a representation of the errors for each class. Each type of sound is well recognized by the convolutional neural network which makes a maximum of 1 or 2 errors while for some classes it appears to perform perfectly. A small difficulty (which however remains only on 5 wrong observations) is found in recognizing the dogs which for example it confuses 3 times for roosters.

B. LSTM of MFCCs

A performing configuration is also the one that uses a long short-term memory layer with 256 units and extends through some dense layers. Also in this case, in addition to the dropouts, a small L1 penalty of 0.001 has been added in the first dense layer to reduce overfitting.

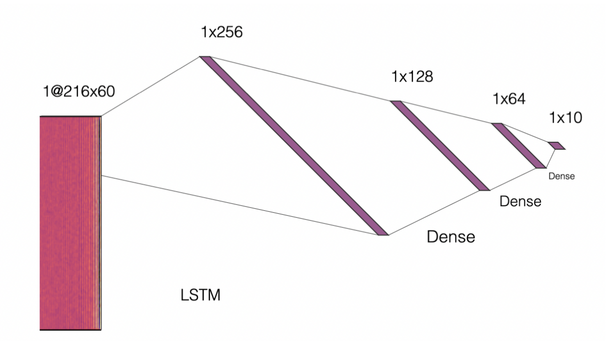


Fig. 10: LSTM architecture

This LSTM has only 366,410 parameters and, as can be seen in the table, also has few layers. However, this simplicity in its configuration does not worsen its performance too much. In fact it will still show good working and classification with high accuracy. But what can be seen in the following plots is that in the training of the model we notice a model slightly less stable than the previous one.

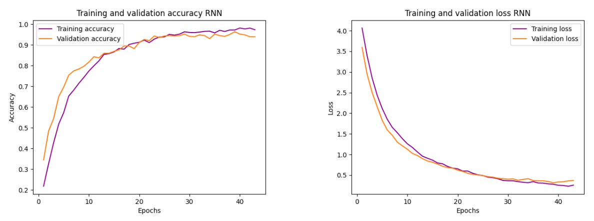


Fig. 11: LSTM with MFCC: train and val accuracies and losses

In this case the early stopping blocked the training after about 40 epochs, a bit more of those used in the previous model. Despite this the time taken is not very different. The trends of the accuracy functions show constant growth with some slight fluctuations. Despite this, the two functions are quite close to each other. This suggests a training process with good generalization skills. The loss functions are instead smoother and this suggests a greater stability. This model also reaches accuracy values around 95 % for validation and around 97 % for train. This shows that it learns very well from train data but without falling into overfitting.
This is also demonstrated by the fact that it is subsequently able to classify well about 95 % of the test set audios.

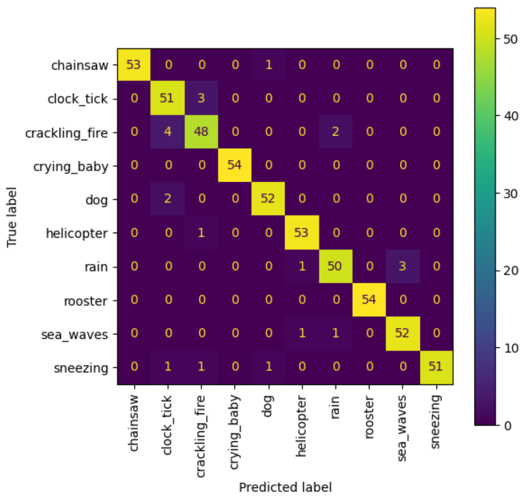


Fig. 12: LSTM with MFCC: confusion matrix

The sound he has the most difficulty recognizing is the crackling of the fire, which it confuses 3 times for the ticking of a clock. Indeed, some audio of fire crackles, sped up by time stretching, are quite ambiguous. If they occur with an almost periodic frequency they could also remind the human ear of a clock ticking.

C. CRNN of MFCCs

The latest model implemented with mfccs to classify the ambient sounds of our dataset is a Convolutional Recurrent Neural Network. As mentioned earlier, this model should perform well and should shows good results because it takes advantage of the strengths of CNN and RNN. In fact, as we see in the table, it has both convolution layers and an LSTM layer. In this case, in addition to the parameters observed in the table, it is necessary to say that L1 penalties of 0.001 have been inserted in the second and third convolutional layers. It is also important to mention padding = ”valid” is applied to the input during the convolution operation in its layers. In the max pooling layers, padding is applied, if necessary, to keep the size of the output equal to the size of the input specifying ’padding=same’.

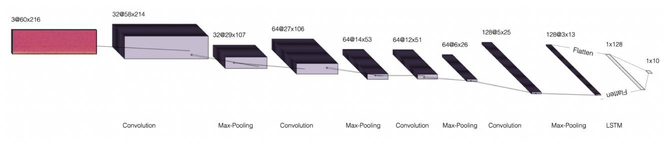


Fig. 13: CRNN for MFCC: model architecture

Among the three, it appears to be the most complex due to the number of layers, but in reality it has fewer parameters (301,674) than the CNN presented in the first subsection. Let’s see how it behaves during training.

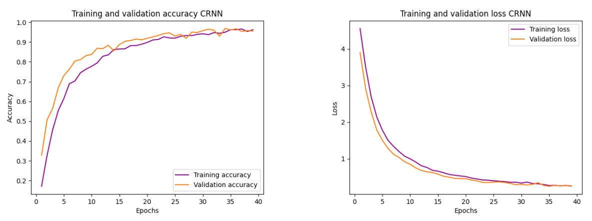
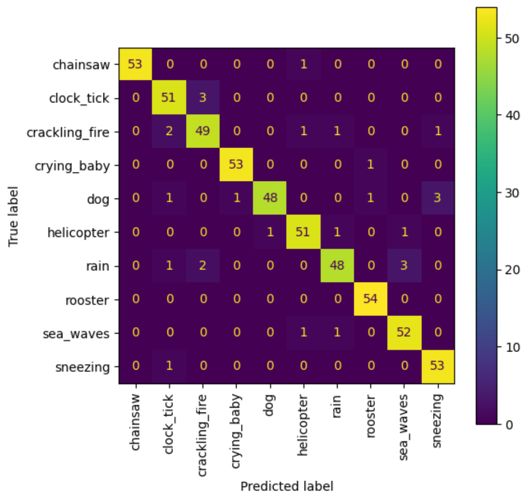


Fig. 14: CRNN: train and val accuracies and losses

As for the CNN, the early stopping in this case stops the training around the 40th epoch. Along this path, both the accuracy functions for train set and validation set are growing and therefore show a gradual improvement in performance over time, but they are not too smooth. In fact, there are several points where there appear to be slight ups and downs that suggest a not perfectly stable model. A gradual and continuous decrease of the loss function can be noted for both datasets without strong oscillations or fluctuations, which in any case are greater for the validation data. As we mentioned before, the model shows more uncertainty, especially compared to the convolutional. At the end of increasing epochs, we reach about 96 % accuracy for the train and about 95 % on the validation. Despite this, as the accuracy function graph suggests, the model could be not the best.



True label \ Predicted label	chainsaw	clock_tick	crackling_fire	crying_baby	dog	helicopter	rain	rooster	sea_waves	sneezing
chainsaw	53	0	0	0	0	1	0	0	0	0
clock_tick	0	51	3	0	0	0	0	0	0	0
crackling_fire	0	2	49	0	0	1	1	0	0	1
crying_baby	0	0	0	53	0	0	0	1	0	0
dog	0	1	0	1	48	0	0	1	0	3
helicopter	0	0	0	0	1	51	1	0	1	0
rain	0	1	2	0	0	0	48	0	3	0
rooster	0	0	0	0	0	0	0	54	0	0
sea_waves	0	0	0	0	0	1	1	0	52	0
sneezing	0	1	0	0	0	0	0	0	0	53

Fig. 15: CRNN with MFCC: confusion matrix

The confusion matrix that results from this classification is obviously good. In fact it shows few errors and no specific confusion between two classes as previously in LSTM. The biggest mistake it makes is confusing three rain audios with those of sea waves and 3 dog audios with the sneezing sound. Looking at the other classes, also in this case we have environmental sounds which are always classified correctly and do not present any errors, and others in which there are 1 or 2.

D. CNN of Spectrograms

After extensively exploring models using MFCC as input, it is now time to adopt a different approach, where the audio will be given as input to the model in the form of a spectrogram. For the model architecture, please refer to Fig. 11.

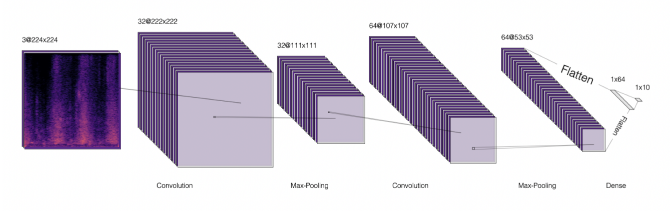


Fig. 16: CNN for Spect: model architecture

We briefly highlight the use of a different input compared to previous approaches, the presence of max-pooling, and the abundant use of dropout to control the model's tendency to overfit (specifically after the two max-pooling layers with dropout rates of 0.4 and 0.5, and after the first Dense layer with a dropout rate of 0.1). Moreover, we chose RELU and softmax as activation functions, adam as the optimizer, and, naturally, categorical crossentropy as the loss function. The model was trained for 80 epochs. The total number of parameters is 257,994.

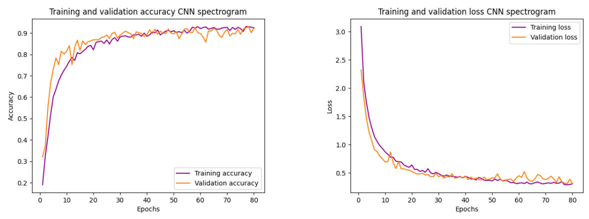


Fig. 17: CNN with spectrograms: train and val accuracies and losses

The good performance of the model can be immediately observed, as it achieves an accuracy of over 93% on test set.

Although the model achieves good overall performance, it is immediately evident from the plots that it is much more unstable compared to the previous ones. In fact, the accuracy and loss scores tend to vary significantly depending on the epochs, and in general, there is a lower robustness of the model. However, we can conclude that this is not a concerning behavior and that the model retains all the characteristics to be considered a good model, without any issues of underfitting or overfitting.

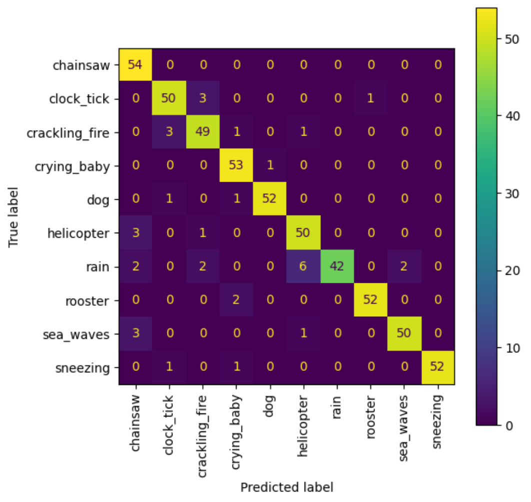


Fig. 18: CNN with spectrograms: confusion matrix

In terms of classification, it is definitely the weakest model, and we can immediately see it from the confusion matrix. It is the only model that struggles to classify an entire class, specifically the rain class. The sound of rain is indeed confused 6 times with that of a helicopter, which is predictable since both are noise sounds. We know that in this classification task, the spectrogram is particularly good at recognizing different types of sounds but lacks the ability to distinguish between two sounds of the same type. This is not a problem for percussive or harmonic sounds that have other sound characteristics that allow for a good separation into classes, but for background sounds, the spectrogram struggles. Overall, however, the model performs very well.

What sets it apart as a slightly weaker model is the execution time and computational complexity. Therefore, in the following paragraph, an algorithmic optimization approach will be analyzed to improve these characteristics of the model.

E. Optimizing Model: The Pruning Approach

At this point, it becomes evident that all the proposed methods have quite good performance. The CNN applied on spectrograms shows a high level of accuracy, potentially attributed to the distinct nature of our classes. However, the key disparity lies in the computation time between the spectrogram approach and models based on MFCC. Specifically, the CNN model on spectrograms is relatively complex compared to other implemented models. Hence, we have chosen to explore the pruning approach to experiment with reducing complexity in terms of computation time and model size without compromising overall performance.

Pruning in neural networks refers to the process of selectively removing connections or neurons from a trained model to improve its efficiency and reduce complexity. By removing unnecessary connections or neurons, we will have a more compact model with different pros:

- fewer computational resources and memory required
- faster inference times
- overfitting prevention
- power saving

In a nutshell, pruning cuts the less important connections and neurons, along with any redundant components of the network. This results in an efficient, compact, lighter model that is less resource-intensive, while maintaining an accuracy close to that of the unpruned model.

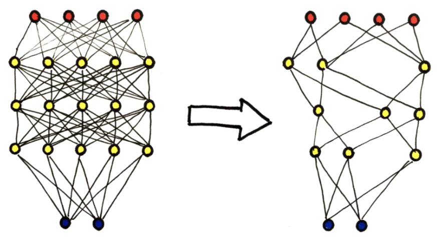


Fig. 19: Pruning mechanism

We have chosen to implement this valuable optimization technique on the least computationally demanding and time-consuming model, namely the CNN for spectrograms. Our results have been highly positive, as:

- Same Accuracy (93.33%)
- Time of Inference: 0.72 (0.85 before the pruning)
- Model Size: 589.68 KB (957.85 KB before the pruning)

We can conclude that pruning is an excellent choice that significantly improves the overall performance of the model.

VII. Comparison with Zohaib Mushtaq and Shun-Feng Su work [3]

We considered appropriate to search for scientific literature that had experimented with a model similar to ours. Our choice fell on the paper 'Environmental sound classification using a regularized deep convolutional neural network with data augmentation' as it is one of the few applications of DL with results on the dataset we selected (ESC-10). Below, we provide a brief description of the model and a brief comparison.

The reference paper conducted experiments with three different types of inputs: mel spectrogram, log-mel spectrogram, and MFCC. Additionally, it implemented the CNN architecture using two different approaches: one including Max-Pooling and the other without it. For the purpose of similarity and direct comparison with our model, we will focus on the CNN model with Max-Pooling trained with MFCC as the input data.

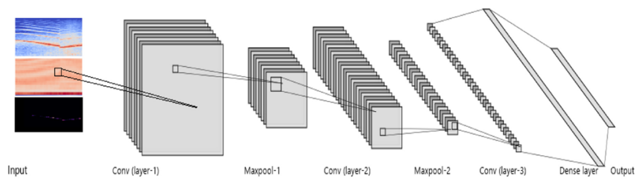


Fig. 20: Paper model structure

It utilizes convolutional layers to extract meaningful features, which are then downsampled using max pooling layers (with pool size of (3,3) and (2,2)). The model has 3 convolutional layers with filter sizes of (5,5), (4,4), and (3,3), respectively, and a stride of (1,1). It also includes dense layers with ReLU activation, with the softmax activation as final dense layer. Dropout regularization (0.5) is applied to prevent overfitting. The model is trained using the Adam optimizer with categorical cross-entropy loss, and its performance is evaluated based on accuracy.

Below we list the main similarities and differences from our model.

Similarities	Differences
CNN architecture	1 less Conv Layer
Use of Drop-out	greater Drop-out (0.5)
Use of Max-Pooling	different pool-sizes
Activation functions	larger filter sizes
Adam and cat-crossentropy	fewer tot. parameters

About performance of the model, we can conclude that the model works well, with an accuracy of 92.9%. This sets it up as a valid alternative, with fewer parameters and slightly less complexity.

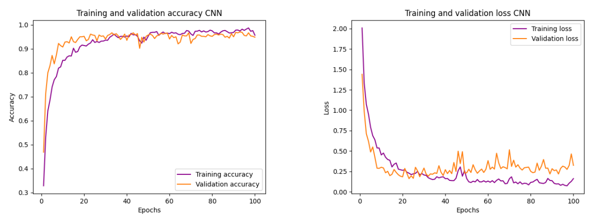


Fig. 21: paper model: train and val accuracies and losses

VIII. Results and comparisons

Input data	Model	Accuracy	Size of model (KB)	Inference time (s)
MFFC	CNN	94.07	1622.69	0.28
MFFC	RNN	95.93	1334.07	0.2
MFFC	CRNN	94.81	1100.95	0.28
Spec	CNN w/o prun	93.33	957.85	0.85
Spec	CNN w/ prun	93.33	589.68	0.72
MFCC	CNN(paper)	92.96	2545.88	0.36

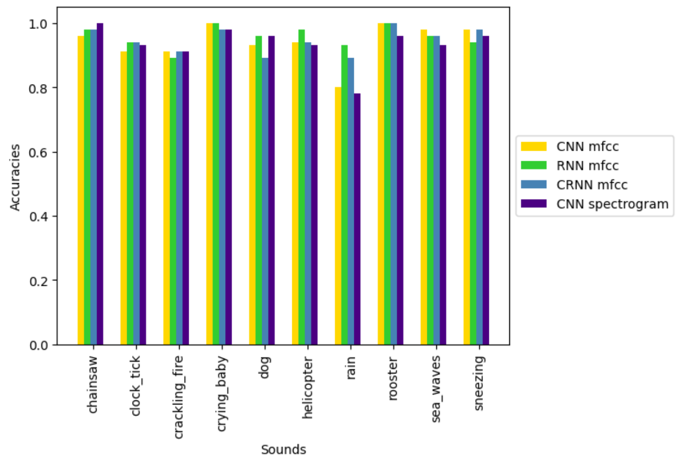


Fig. 22: Models performances on different audio class

IX. Conclusion

After experimenting with different approaches and models, we have reached some final conclusions:

The most effective approach we found was using MFCC (Mel Frequency Cepstral Coefficients). It has proven to be capable of capturing all the necessary information for accurate classification. However, the spectrogram approach also performed well. This can be attributed to the distinct nature of the ten classes, which include melodic, rhythmic, and other variations. Thus, even the spectrogram was able to extract relevant information.

All the models we tested yielded satisfactory results. However, the RNN (Recurrent Neural Network) model emerged as the top performer. It achieved an impressive accuracy rate of over 96%.

Our findings align with existing publications on our reference dataset ESC, where the use of MFCC has been recognized as a successful approach, and the most commonly proposed models for the task are CNN and RNN.

In conclusion, our results validate the effectiveness of MFCC and RNN for our specific classification task.

References

[1] K. J. Piczak, "Esc: Dataset for environmental sound classification," in *Proceedings of the 23rd ACM international conference on Multimedia*, pp. 1015–1018, 2015.

[2] 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.

[3] Z. Mushtaq and S.-F. Su, "Environmental sound classification using a regularized deep convolutional neural network with data augmentation," *Applied Acoustics*, vol. 167, p. 107389, 2020.